Document Title	Specification of Diagnostics
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	723

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Document quality improvement, clarifications and fixing bugs • Document structure updated • Formalized generated interface classes for DID, RID and DataElements • Standardized Violations added • Term Reentrancy is changed to Concurrency • Support DoIP amendment 2023 protocol version 4 • Harmonization with CP • Explicit no-debouncing for ara::diag::monitor • SecurityEvents added
2023-11-23	R23-11	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • SOVD Concept Part 2 implemented • Service 0x29 refinements

△

2022-11-24	R22-11	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Introduced DTC suppressed feature • Standardize mapping of vendor specific error codes to UDS Error codes • Introduced 0x38 RequestFileTransfer • Introduced SOVD Concept
2021-11-25	R21-11	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Introduced UDS service 29 • Introduced Event Combination in chapter 7
2020-11-30	R20-11	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Validated requirements from concept DoIPExtension • Introduced UDS services 2A & 2C
2019-11-28	R19-11	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Partly removed obsolete requirements • Removed obsolete service interfaces • Changed Document Status from Final to published

▽

△

2019-03-29	19-03	AUTOSAR Release Management	• Document quality improvement and fixing bugs • Introduced ara::diag interfaces in draft state
2018-10-31	18-10	AUTOSAR Release Management	• Diagnostic Protocol replaced by Diagnostic Conversations • ResponseOnEvent, CommunicationControl, EcuReset added • Chapter 7 overall rework and updates • Chapter 8 split into chapter 8 (C++ API) and chapter 9 (Service Interfaces)
2018-03-29	18-03	AUTOSAR Release Management	• Chapter 7.1. Software Cluster added • Chapter 7.2. Diagnostic Service Management, common parts for all services separated • Chapter 7.3. Event Management, several additions and rework • Chapter 8. API specification, complete rework
2017-10-27	17-10	AUTOSAR Release Management	• General API rework • TP Plug-in interface • Introduction of SoftwareCluster in APIs • Additional UDS services like SecurityAccess
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Adaptive Diagnostic Management (DM).

The DM is a diagnostic server implementation that realizes

- a UDS server instance according to ISO 14229-1[1]
- SOVD according to ASAM for the AUTOSAR Adaptive Platform.

As transport layer for UDS a flexible concept is applied. DoIP protocol based on ISO 13400-2[2] or a custom implementation o a transport protocol can be used.

For SOVD the HTTP transport protocol with REST services is used.

1.1 Diagnostic interface

Since release R19-03 a C++ interface was introduced for diagnostics as a replacement for the former ara::com based service interface.

1.2 AUTOSAR Diagnostic Extract Template (DEXT)

The AUTOSAR Diagnostic Extract Template (DEXT) [3] is the configuration input to the DM.

1.3 Software Cluster

The AUTOSAR adaptive platform is able to be extended with new software packages without re-flashing the entire ECU. The individual software packages are described by SoftwareClusters. To support the current approaches of diagnostic management (like software updates), each SoftwareCluster have its own DiagnosticAddresses.

The DM is intended to support an own diagnostic server instance per installed SoftwareCluster. All diagnostic server instances share a single TransportLayer instance (e.g. DoIP on TCP/IP port 13400).

1.3.1 Diagnostic Server

The Diagnostic Communication Management response handling basically resembles the functionality of the Dcm BSW module of the AUTOSAR Classic platform. I.e. it is responsible for processing/dispatching of diagnostic services according to ISO

14229-1[1] and SOVD services as per Service-Oriented Vehicle Diagnostics standard Specification [4]. That means:

- Receiving diagnostic request messages from the network layer.
  - UDS DoIP or proprietary transport protocol requests
  - SOVD HTTP/REST requests
- Processing and extracting information from the request
  - transport layer independent UDS information
  - entity path and resource information for SOVD requests
- Dispatching the request towards the Diagnostic Server instances depending on
  - the target address and target address type (physical or functional) for UDS requests
  - the addressed entity and its path in the URI for SOVD requests
- Managing multiple diagnostic requests and clients by
  - correlating the diagnostic request to an existing UDS session (if already exists).
  - handling modes and locks for the SOVD resource requested by the SOVD client and protect access to the resources (e.g., operations).
- Checking whether the diagnostic request is allowed based on
  - current session and security settings for UDS.
  - authorization and validation of SOVD client request to grant access to the requested resources
- If diagnostic request is NOT allowed
  - generate negative UDS response and send it to the network layer for UDS requests
  - generate corresponding SOVD error codes and send it as HTTP error responses for SOVD requests.
- If diagnostic (UDS and SOVD) request is allowed, depending on DM's configuration and request type,
  - either process the service internally within Diagnostic Communication Management function block of DM
  - or process the service internally within Diagnostic Event Management function block of DM
  - or hand it over for processing to an (external to DM) Adaptive Application

The figure below depicts those processing steps and functional blocks of DM's Diagnostic Communication Management part.

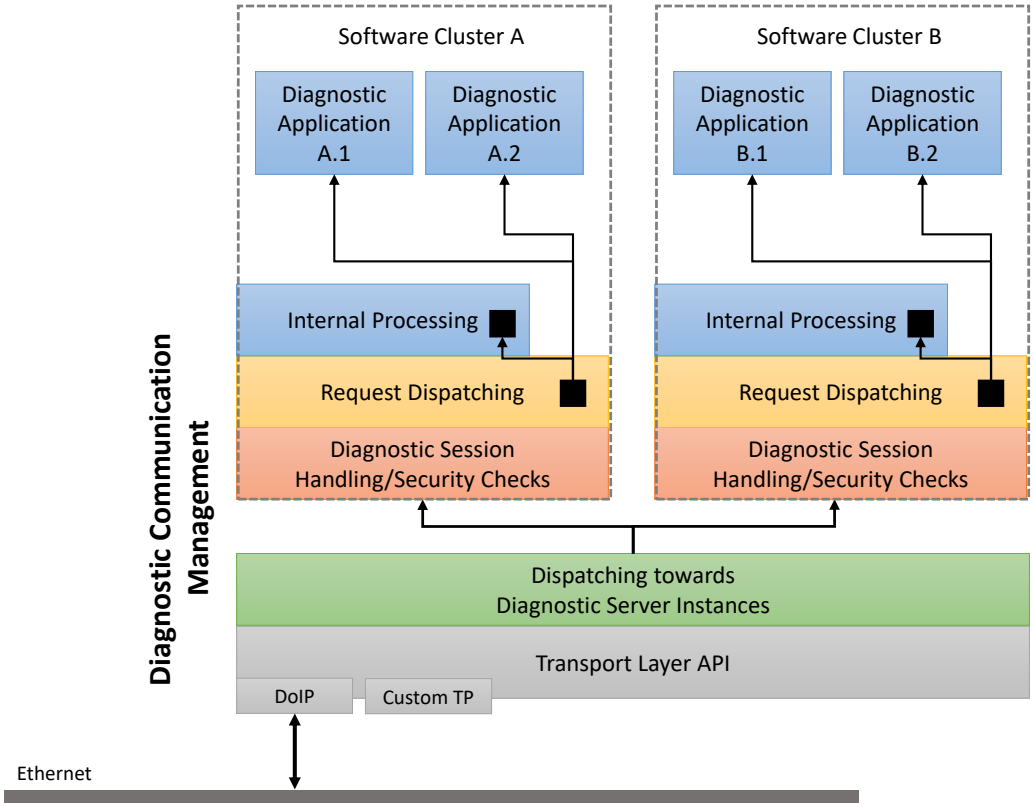

Figure 1.1: Architecture Diagnostic Communication Management

1.3.2 Diagnostic Managers external dependencies

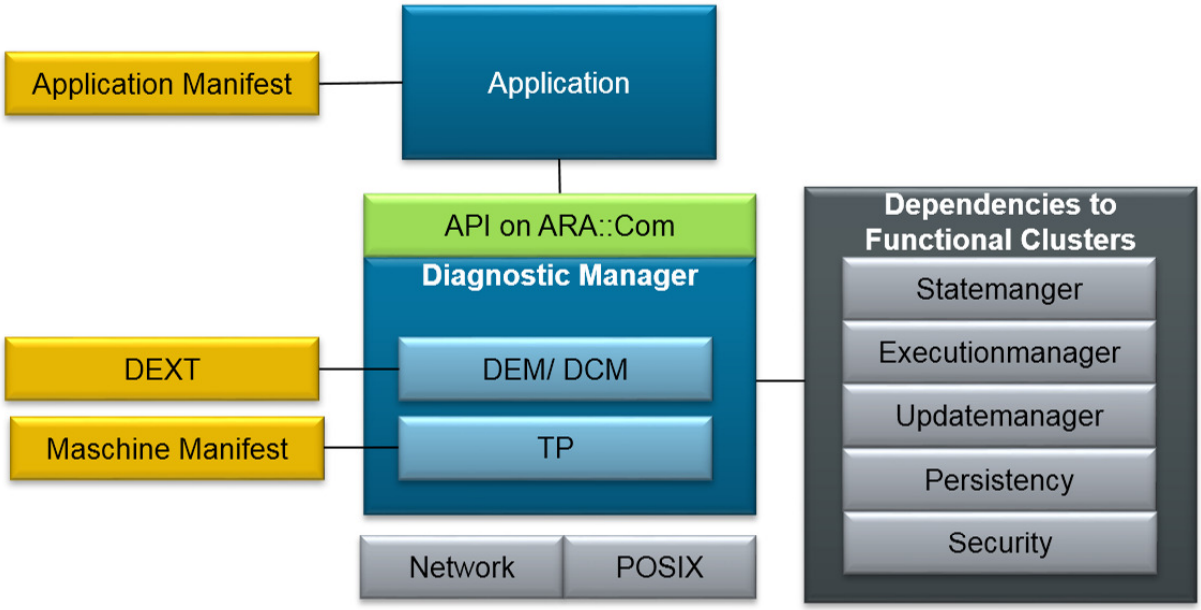

Figure 1.2: Diagnostic Managers external dependencies

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations that are only relevant within this specification. A general list of acronyms and abbreviations is available in [5].

Abbreviation / Acronym:	Description:
AA	AUTOSAR Adaptive Application
AP	AUTOSAR Adaptive Platform
API	Application Programming Interface
ASAM	Association for Standardization of Automation and Measuring Systems
BSW	Basic Software
Channel	An abstraction of a network specific communication channel. In CAN networks a Channel can be identified via CAN identifier. In Ethernet networks a Channel might be defined by the quadruple Src-IP, Src-Port, Target-IP, Target-Port.
CP	AUTOSAR Classic Platform
Dcm	Diagnostic Communication Manager (Module of the AUTOSAR Classic Platform)
DDDID	Dynamically Defined Data Identifier according to ISO 14229-1[1].
DEXT	AUTOSAR Diagnostic Extract[3], describing diagnostic configuration of an ECU
DID	Data Identifier according to ISO 14229-1[1]. This 16 bit value uniquely defines one ore more data elements (parameters) that can are used in diagnostics to read, write or control data.
DM	AUTOSAR Adaptive Diagnostic Management
DNS	Domain Name System
DoIP	Diagnostics over Internet Protocol (Communication protocol of automotive electronics according to ISO-13400-2[2])
DTC	Diagnostic Trouble Code according to ISO 14229-1[1]
ECU	Electronic control unit
EDR	`Extended Data Record`
EID	Entity Identification, as used in DoIP specification
Execution Management	Functional cluster Execution Management
FDC	Fault Detection Counter according to 14229-1[1]. Has always the value range from -128 = FDC_{min} = "`FinallyHealed`" to +127 = FDC_{max} = "`FinallyDefective`"
GID	Group identifier as used in DoIP
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
MetaInfo	Meta-Information in the form of a key-value map, which is given from DM to external service processors.
NRC	Negative Response Code used by UDS in the diagnostic response to indicate the tester that a certain failure has occurred and the diagnostic request was not processed.
OBD	"On-Board Diagnostics"; Generally: A vehicles ability for self diagnosis and reporting to external test tools. Specifically here, the protocol is meant, as defined in SAE J1979, ISO 15031, ISO 27145 and others.
OEM	"Original Equipment Manufacturer", but in this document herein, it is used for "Vehicle Manufacturer".
PDID	Periodic Data Identifier according to ISO 14229-1[1].

Abbreviation / Acronym:	Description:
PowerMode	Vehicle basic status information retrieval of DoIP
REST	Representational State Transfer
RoE	Response on Event
SA	SourceAddress of a UDS request
SDG	Special Data Group
SID	Service Identifier, identifying a diagnostic service according to UDS, such as 0x14 ClearDiagnosticInformation
SOVD	Service-Oriented Vehicle Diagnostics
TA	TargetAddress of a UDS request
TLS	Transport Layer Security
UDS	Unified Diagnostic Services
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VIN	Vehicle Identification Number according to ISO-3779

Terms:	Description:
Aging	Unlearning/deleting of a no longer failed event/DTC after a defined number of operation cycles from event memory.
Diagnostic Client	A Diagnostic Client is a diagnostic service requester, i.e. sends a UDS request to the Diagnostic Server. Usually the Diagnostic Client is an external tester equipment but can also be another vehicle internal ECU.
Diagnostic Communication Management	Diagnostic Communication Management is the part of the Diagnostic Management which belongs to tester communication and the processing of UDS services.
Diagnostic Conversation	Diagnostic Conversation represents a conversation between Diagnostic Client (Tester) and Diagnostic Server.
Diagnostic Event Management	Diagnostic Event Management is the part of the Diagnostic Management which belongs to processing and storing of diagnostic events and associated data.
Diagnostic Management	Diagnostic Management is a placeholder for the complete functionality of diagnostic communication and event handling.
Diagnostic Server instance	Diagnostic Server (DM) is intended to support an own Diagnostic Server instance per installed *SoftwareCluster*, see section 7.3 for a detailed description. Each of those Server instances has and manages its own resources and is responsible for dispatching and processing of diagnostic services.
Diagnostic Service instance	A diagnostic service instance implements a concrete use of a diagnostic service in a given context. It refers to a DiagnosticServiceClass and the DiagnosticAccessPermission , see 7.3.2.5.3 for a detailed description.
Displacement	In case of an Event memory overflow: Replacing the most insignificant stored event memory entry by a reported event which needs to be stored and is more significant.
DTC group	Uniquely identifies a set of DTCs. A DTC group is mapped to the range of valid DTCs. By providing a group of DTCs it is expressed that a certain operation is requested on all DTCs of that group. The DTC group definition is provided by ISO 14229-1[1] and OEM/supplier-specific.
DTCStatusAvailabilityMask	The DTCStatusAvailabilityMask - byte is used in UDS responses to requests for certain sub-functions of service 0x19. It express which of the UDS DTC status bits are supported by the DM for masking purposes.

Terms:	Description:
Enable Conditions	The criteria / conditions under which the test results from the monitors in the AA's are valid and shall be processed by DM. Configuration is done per event.
Extended Data Records	Contains statistical data for a DTC. Extended data records are assigned to DTCs and maintained and stored by the DM.
Event	An *event* (also *diagnostic event*) uniquely identifies a fault path of the system. An application monitors the system and reports events to the DM.
Event memory	The DM stores information about events in the event memory. There can be multiple event memories, each keeping information independently from each other. Examples of the event memory is the UDS primary event memory or the up to 256 user-defined event memories.
Event memory overflow	An event memory overflow occurs, if this specific event memory is full and the next event occurs to be stored in this event memory.
Event status	Bit-packed status information based on Event level. Contains the following bits: • Nr. Definition: • 0 testFailed • 1 testFailedThisOperationCycle • 6 testNotCompletedThisOperationCycle Compare UDS DTC status bit
Fail-safe reaction	(Sometimes also called "limp home mode"): Reaction to avoid or minimize harm or damage in case of a failure.
GroupOfAllDTCs	Identifies a special DTC group that contains all DTCs. This DTC group is identified by the DTC value 0xFFFFFF in 14229-1[1] and contains by default all DTCs of a fault memory. It is present by default in the DM and requires no configuration.
Internal, External	Classifies if a DiagnosticDataElement is either managed internally inside DM or by an external adaptive applications, see 7.3.6.1 for the precise definition.
Internally, Externally	Definition of the support type of a SID by the DM. Internally means processing is done by DM itself, Externally means an external service processor is used.
internal data element	A DiagnosticDataElement which is provided by the DM itself. See also 7.3.6.1.
Monitor	A *monitor* (also *diagnostic monitor*) is a piece of software running within an application, monitoring the correct functionality of a certain system part. The result of such a function check is reported to the DM in form of a diagnostic event.
Operation cycle	A new operation cycle is the start of a new monitoring cycle. This is reflected in a reset of the testFailedThisOperationCycle and testNotCompletedThisOperationCycle bits in the DTC status and optional notification to the monitor to restart the monitoring.
OpenAPI specification	Specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services.
Primary event memory	The primary event memory is used to store events and event related data. It is typically used by OEMs for after sales purposes, containing information to repair the vehicle.

Terms:	Description:
Snapshot Record	Snapshots (sometimes referred to as freeze frames) are specific data records associated with a DTC and stored in the fault memory at a certain point of time during fault detection. The DTC specific data-parameters are intended to ease the fault isolation process.
SoftwareCluster	A SoftwareCluster groups all AUTOSAR artifacts which are relevant to deploy software on a machine. This includes the definition of applications, i.e. their executables, application manifests, communication and diagnostics. In the context of diagnostics a SoftwareCluster can be addressed individually by its own set of diagnostic addresses.
SourceAddress	A Source Address is used to encode client and server identifiers. In a UDS request the source address encodes the Diagnostic Client whereas the source address in a UDS response encodes the Diagnostic Server.
SOVD entity	Entity in context of SOVD[4]
SOVD lock	SOVD locking mechanism as defined by [4]
TargetAddress	A Target Address is used to encode client and server identifiers. In a UDS request the target address encodes the Diagnostic Server whereas the target address in a UDS response encodes the Diagnostic Client.
Transport Protocol Handler	A subcomponent of DM implementing a particular Transport Protocol (either DoIP or any other UDS Transport Layer).
Transport Protocol Manager	Link between UDS Transport Layer and Application Layer.
UDS service	A diagnostic service as defined in ISO 14229-1[1].
UDS DTC status bit	UDS DTC status bit as defined in ISO 14229-1[1] Annex D.2; Each single bit position represents and documents a certain status information for the connected DTC. The following eight bits are defined: **Nr:** Definition: **0** testFailed **1** testFailedThisOperationCycle **2** pendingDTC **3** confirmedDTC **4** testNotCompletedSinceLastClear **5** testFailedSinceLastClear **6** testNotCompletedThisOperationCycle **7** warningIndicatorRequested All eight bits constitute the UDS DTC status byte. Compare Event status
UDS DTC status byte	Bit-packed DTC status information byte as defined in ISO 14229-1[1], based on DTC level. Contains the UDS DTC status bits.
User-defined event memory	The user-defined event/fault memory is used by the UDS service 0x19 with subfunctions 0x17, 0x18 and 0x19. It behaves as the primary event memory but contains data independent from the primary fault memory. It is used to store information that are relevant for different purposes such as warranty or development.

Terms:	Description:
Non-volatile Memory	In the context of DM, Non-volatile Memory refers to the persistent information over the shutdown of the DM process. This does not depend on HW details.

3 Related documentation

3.1 Input documents & related standards and norms

[1] ISO 14229-1(2020) – Unified diagnostic services (UDS) – Part 1: Application layer (Release 2020-02)
https://www.iso.org

[2] ISO 13400-2:2019 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Release 2019-12)
https://www.iso.org/standard/74785.html

[3] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

[4] ASAM SOVD Service-Oriented Vehicle Diagnostics - API Specification V1.0.0
http://www.asam.net

[5] Glossary
AUTOSAR_FO_TR_Glossary

[6] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[7] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[8] Requirements on Diagnostics
AUTOSAR_FO_RS_Diagnostics

[9] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[10] ISO 15765-2 – Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2: Network layer services

[11] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[12] ISO 14229-2(2021) – Unified diagnostic services (UDS) - Part 2: Session layer services (Release 2021-10)
https://www.iso.org

[13] Requirements on Intrusion Detection System
AUTOSAR_FO_RS_IntrusionDetectionSystem

[14] Specification of Language Binding for modeled AP data types
AUTOSAR_AP_SWS_LanguageBindingForModeledAPdatatypes

[15] Specification of Platform Types for Adaptive Platform
AUTOSAR_AP_SWS_PlatformTypes

[16] ISO/IEC 14882:2014, Information technology – Programming languages – C++
https://www.iso.org

[17] ISO 14229-1(2013) – Unified diagnostic services (UDS) – Part 1: Application layer (Release 2013-03)
https://www.iso.org

3.2 Further applicable specification

AUTOSAR provides a core specification [6] which is also applicable for this functional cluster. The chapter "General requirements for all Functional Clusters" of [6] shall be considered an additional and required specification for implementing this functional cluster.

4 Constraints and assumptions

4.1 Known Limitations

This chapter describes known limitation of the DM in respect to general claimed goals of the module. The nature of constraints can be a general exclusion of a certain domain / functionality or it can be that the provided standard has not yet integrated this functionality and will do so in future releases.

- Only scheduler type 1 from [1] is supported for service 0x2A
- Subfunction defineByMemoryAddress for service 0x2C is not supported
- OBD ISO 15031 and WWH OBD ISO 27145 is not supported by the DM.
- *Software Cluster*/*Diagnostic Server instances* are supported by DM interfaces but are not specified in detail.
- *DoIP edge node* is not supported by the DM.
- The following UDS services are not implemented by the DM:
  - 0x23 ReadMemoryByAddress
  - 0x24 ReadScalingDataByIdentifier
  - 0x2F InputOutputControlByIdentifier
  - 0x3D WriteMemoryByAddress
  - 0x83 AccessTimingParameter
  - 0x84 SecuredDataTransmission
  - 0x87 LinkControl
- Sub-functions of UDS services are implemented according to ISO 14229-1[1] unless explicitly stated.
- The UDS mirror event memory is not supported by the DM. As a result of this, the DM does not support the UDS service
  - 0x19 with subfunction 0x0F (reportMirrorMemoryDTCByStatusMask)
  - 0x19 with subfunction 0x10 (reportMirrorMemoryDTCExtDataRecordByDTCNumber)
  - 0x19 with subfunction 0x11 (reportNumberOfMirrorMemoryDTCByStatusMask)
- The OBD/WWH OBD is not supported by the DM. As a result of this, the DM does not support the UDS service
  - 0x19 with subfunction 0x05 (reportDTCStoredDataByRecordNumber)

  - 0x19 with subfunction 0x12 (reportNumberOfEmissionsOBDDTCByStatusMask)
  - 0x19 with subfunction 0x13 (reportEmissionsOBDDTCByStatusMask)
  - 0x19 with subfunction 0x42 (reportWWHOBDDTCByMaskRecord)
  - 0x19 with subfunction 0x55 (reportWWHOBDDTCWithPermanentStatus)
- The following general UDS services of ReadDTCInformation are not supported:
  - 0x19 with subfunction 0x08 (reportDTCBySeverityMaskRecord)
  - 0x19 with subfunction 0x09 (reportSeverityInformationOfDTC)
  - 0x19 with subfunction 0x0B (reportFirstTestFailedDTC)
  - 0x19 with subfunction 0x0C (reportFirstConfirmedDTC)
  - 0x19 with subfunction 0x0D (reportMostRecentTestFailedDTC)
  - 0x19 with subfunction 0x0E (reportMostRecentConfirmedDTC)
  - 0x19 with subfunction 0x15 (reportDTCWithPermanentStatus)
  - 0x19 with subfunction 0x16 (reportDTCExtDataRecordByRecordNumber)
- Event Memory: Variant handling at runtime for events/DTCs is not supported.
- Event Memory: Details for combined events are not specified.
- Persistent Storage of failed attempts to change security level : After each increment of the attempt counter, it shall be persisted to survive accidental or intended resets. Here the option to select the persistent storage is mandatory in Adaptive Autosar.
- Only Subfunction 0x01 (ON) and Subfunction 0x02 (OFF) is supported for service 0x85.
- For the UDS service 0x86 ResponseOnEvent the following applies:
  - Queuing of events is not supported.
  - Regarding the request message parameter eventWindowTime (refer to B.2 in [1]), the only values supported are infiniteTimeToResponse and powerWindowTime.
- The Diagnostic Manager only implements the UDS Service Authentication (0x29) via PKI certificate exchange. Authentication with challenge-response (ACR) is currently out of scope of the Diagnostic Manager.
- Manufacturer and Supplier specific service checks according to "Figure 6 - General server response behavior for request messages with SubFunction parameter" of ISO 14229-1[1] are not supported.

- OEM-specific error codes are by intention not part of the AUTOSAR standardization but can be added platform vendor specific as an additional ErrorDomain.
- The Dcm does not support the behavior described in the ISO 14229-1[1] chapter 10.4.that after a powerup a single false access attempt in the previous power cycle already starts the security delay time. ([SWS_DM_00479] is followed instead.)

For SOVD[4] the following limitations apply:

- The SOVD entities area, app and function are not supported.
- Custom SOVD modes are not supported. Only standardized SOVD modes communication_control (section 7.6.3.5.7) and control_dtc_settings (section 7.6.3.5.8) are supported
- For executions of SOVD operations custom capabilites are not supported.
- For SOVD operations the attribute modes is not supported.
- SOVD locks cannot be acquired on SOVD component (Machine) level. Acquiring SOVD locks is only possible on SOVD sub-component (SoftwareCluster) level.
- Value 0x00 is not supported as default value for EID,GID and VIN as specified in ISO 13400-2[2]

4.1.1 AP stablization

Thread-safety for some APIs is not defined.

5 Dependencies to other Functional Clusters

This chapter defines the dependencies of this functional cluster to other functional clusters. AUTOSAR decided not to standardize interfaces which are exclusively used between functional clusters to allow efficient implementations which might depend e.g., on the used operating system. The goal of this chapter is to provide an informative guideline for the interactions between functional clusters without specifying syntactical details. This ensures compatibility between documents specifying different functional clusters and supports parallel implementation of different functional clusters. Details of internal interfaces are up to the platform provider. Additional internal interfaces, parameters, and return values can be added. A detailed technical architecture documentation of the overall AUTOSAR Adaptive Platform is provided in [7].

5.1 Provided Interfaces

This section provides an overview of the public interfaces provided by this functional cluster towards other functional clusters.

Interface	*Functional Cluster*	*Purpose*
No provided interfaces		

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

This section provides an overview of the public interfaces required by this functional cluster from other functional clusters.

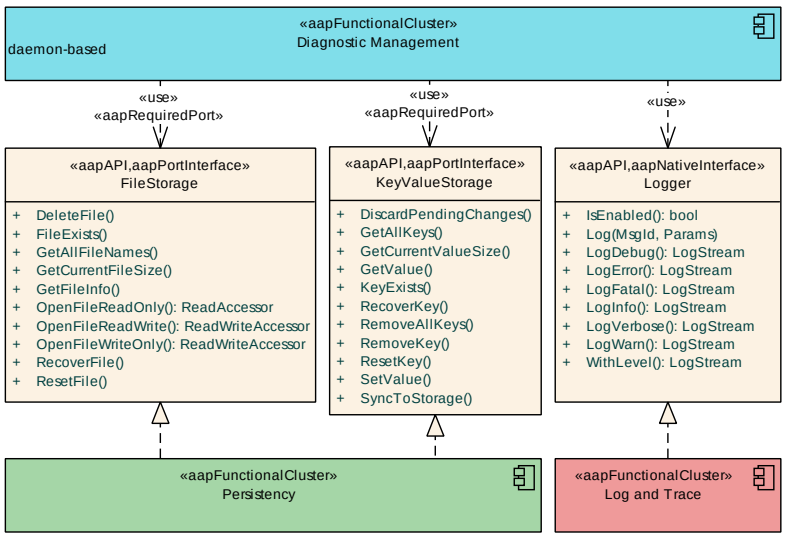

Figure 5.1: Interfaces required by Diagnostic Management

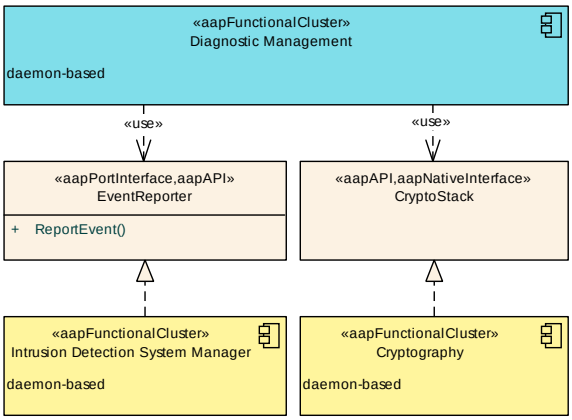

Figure 5.2: Interface required by Diagnostic Management from Adaptive Intrustion Detection System Manager

Figures 5.1, 5.2 show the interfaces required by `Diagnostic Management` from other Functional Clusters within the AUTOSAR Adaptive Platform. Table 5.2 provides a complete list of required interfaces from other Functional Clusters within the AUTOSAR Adaptive Platform.

Functional Cluster	***Interface***	***Purpose***
Cryptography	CryptoStack	This interface may be used e.g., to access keys for secure diagnostics.
Execution Management	ExecutionClient	This interface is used to report the status of the `Diagnostic Management` daemon process(es).
Intrusion Detection System Manager	EventReporter	`Diagnostic Management` uses this interface to report standardized security events.
Log and Trace	Logger	`Diagnostic Management` shall use this interface to log standardized messages.
Persistency	FileStorage	Used to store associated data of diagnostic trouble codes (e.g., freeze frames).
Persistency	KeyValueStorage	Used to store properties of diagnostic trouble codes and diagnostic sessions.
Platform Health Management	SupervisedEntity	`Diagnostic Management` should use this interface to enable supervision of its daemon process(es) by `Platform Health Management`.

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in Diagnostics [8] and the AUTOSAR RS General [9], 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
[RS_AP_00114]	C++ interface shall be compatible with C++14	[SWS_DM_00579] [SWS_DM_00580] [SWS_DM_00581] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00596] [SWS_DM_00598] [SWS_DM_00640] [SWS_DM_80001] [SWS_DM_80002] [SWS_DM_80011] [SWS_DM_80012] [SWS_DM_80021] [SWS_DM_80022]
[RS_AP_00119]	Return values / application errors	[SWS_DM_00514] [SWS_DM_00515] [SWS_DM_00516] [SWS_DM_00517] [SWS_DM_00518] [SWS_DM_00519] [SWS_DM_00520] [SWS_DM_00521] [SWS_DM_00522] [SWS_DM_00523] [SWS_DM_00524] [SWS_DM_00525] [SWS_DM_00526] [SWS_DM_00527] [SWS_DM_00528] [SWS_DM_00529] [SWS_DM_00530] [SWS_DM_00531] [SWS_DM_00532] [SWS_DM_00533] [SWS_DM_00534] [SWS_DM_00535] [SWS_DM_00536] [SWS_DM_00537] [SWS_DM_00543] [SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00556] [SWS_DM_00557] [SWS_DM_00559] [SWS_DM_00560] [SWS_DM_00594] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00618] [SWS_DM_00619] [SWS_DM_00636] [SWS_DM_00637] [SWS_DM_00638] [SWS_DM_00653] [SWS_DM_00671] [SWS_DM_00725] [SWS_DM_00735] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00766] [SWS_DM_00774] [SWS_DM_00776] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00792] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00802] [SWS_DM_00808] [SWS_DM_00809] [SWS_DM_00836] [SWS_DM_00985] [SWS_DM_00986] [SWS_DM_00987] [SWS_DM_00988] [SWS_DM_00989] [SWS_DM_00990] [SWS_DM_00991] [SWS_DM_00992] [SWS_DM_00993] [SWS_DM_00994] [SWS_DM_00995] [SWS_DM_00996] [SWS_DM_00997] [SWS_DM_00998] [SWS_DM_00999] [SWS_DM_01000] [SWS_DM_01001] [SWS_DM_01002] [SWS_DM_01003] [SWS_DM_01004] [SWS_DM_01005] [SWS_DM_01006] [SWS_DM_01016] [SWS_DM_01088] [SWS_DM_01367] [SWS_DM_01698] [SWS_DM_01699] [SWS_DM_01709] [SWS_DM_01806] [SWS_DM_01807] [SWS_DM_01808]

△

Requirement	Description	Satisfied by
		△ [SWS_DM_01809] [SWS_DM_01810] [SWS_DM_01811] [SWS_DM_01812] [SWS_DM_01813] [SWS_DM_01814] [SWS_DM_01815] [SWS_DM_01816] [SWS_DM_01817]
[RS_AP_00120]	Method and Function names	[SWS_DM_00309] [SWS_DM_00313] [SWS_DM_00314]
[RS_AP_00121]	Parameter names	[SWS_DM_00309] [SWS_DM_00313] [SWS_DM_00314] [SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_01695] [SWS_DM_01696] [SWS_DM_01697] [SWS_DM_01702] [SWS_DM_01972] [SWS_DM_01973]
[RS_AP_00122]	Type names	[SWS_DM_00309] [SWS_DM_00313] [SWS_DM_00314]
[RS_AP_00125]	Enumerator and constant names	[SWS_DM_00514] [SWS_DM_00526] [SWS_DM_00559] [SWS_DM_00560] [SWS_DM_00642] [SWS_DM_00643] [SWS_DM_00645] [SWS_DM_00690] [SWS_DM_00705] [SWS_DM_00706] [SWS_DM_00710] [SWS_DM_01004] [SWS_DM_01278] [SWS_DM_01279] [SWS_DM_01280] [SWS_DM_01281] [SWS_DM_01282] [SWS_DM_01283] [SWS_DM_01284] [SWS_DM_01285] [SWS_DM_01286] [SWS_DM_01806]
[RS_AP_00127]	Usage of ara::core types	[SWS_DM_00579] [SWS_DM_00580] [SWS_DM_00581] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00596] [SWS_DM_00598] [SWS_DM_00640]
[RS_AP_00128]	Error reporting	[SWS_DM_00579] [SWS_DM_00580] [SWS_DM_00581] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00596] [SWS_DM_00598] [SWS_DM_00640] [SWS_DM_01126] [SWS_DM_01127] [SWS_DM_01128]
[RS_AP_00132]	noexcept behavior of API functions	[SWS_DM_00515] [SWS_DM_00516] [SWS_DM_00517] [SWS_DM_00518] [SWS_DM_00519] [SWS_DM_00520] [SWS_DM_00521] [SWS_DM_00522] [SWS_DM_00523] [SWS_DM_00524] [SWS_DM_00525] [SWS_DM_00527] [SWS_DM_00528] [SWS_DM_00529] [SWS_DM_00530] [SWS_DM_00531] [SWS_DM_00532] [SWS_DM_00533] [SWS_DM_00534] [SWS_DM_00535] [SWS_DM_00536] [SWS_DM_00537] [SWS_DM_00985] [SWS_DM_00986] [SWS_DM_00987] [SWS_DM_00988] [SWS_DM_00989] [SWS_DM_00990] [SWS_DM_00991] [SWS_DM_00992] [SWS_DM_00993] [SWS_DM_00994] [SWS_DM_00995] [SWS_DM_00996] [SWS_DM_00997] [SWS_DM_00998] [SWS_DM_00999] [SWS_DM_01000] [SWS_DM_01001] [SWS_DM_01002] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_DM_01003] [SWS_DM_01004] [SWS_DM_01005] [SWS_DM_01006] [SWS_DM_01807] [SWS_DM_01808] [SWS_DM_01809] [SWS_DM_01810] [SWS_DM_01811] [SWS_DM_01812] [SWS_DM_01813] [SWS_DM_01814] [SWS_DM_01815] [SWS_DM_01816] [SWS_DM_01817]
[RS_AP_00133]	noexcept behavior of move and swap operations	[SWS_DM_00610] [SWS_DM_00612]
[RS_AP_00134]	noexcept behavior of class destructors	[SWS_DM_00553] [SWS_DM_00584] [SWS_DM_00586] [SWS_DM_00588] [SWS_DM_00590] [SWS_DM_00635] [SWS_DM_00648] [SWS_DM_00665] [SWS_DM_00713] [SWS_DM_00723] [SWS_DM_00733] [SWS_DM_00743] [SWS_DM_00753] [SWS_DM_00763] [SWS_DM_00773] [SWS_DM_00788] [SWS_DM_00798] [SWS_DM_00807] [SWS_DM_00832] [SWS_DM_01365] [SWS_DM_01524] [SWS_DM_01706] [SWS_DM_01737]
[RS_AP_00137]	Connecting run-time interface with model	[SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00552] [SWS_DM_00585] [SWS_DM_00587] [SWS_DM_00589] [SWS_DM_00616] [SWS_DM_00631] [SWS_DM_00633] [SWS_DM_00634] [SWS_DM_00647] [SWS_DM_00664] [SWS_DM_00712] [SWS_DM_00722] [SWS_DM_00732] [SWS_DM_00742] [SWS_DM_00752] [SWS_DM_00762] [SWS_DM_00772] [SWS_DM_00787] [SWS_DM_00797] [SWS_DM_00806] [SWS_DM_01364] [SWS_DM_01695] [SWS_DM_01696] [SWS_DM_01697] [SWS_DM_01702] [SWS_DM_01705] [SWS_DM_01727] [SWS_DM_01972] [SWS_DM_01973]
[RS_AP_00138]	Return type of asynchronous function calls	[SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00557] [SWS_DM_00579] [SWS_DM_00580] [SWS_DM_00581] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00596] [SWS_DM_00598] [SWS_DM_00618] [SWS_DM_00636] [SWS_DM_00637] [SWS_DM_00640] [SWS_DM_00724] [SWS_DM_00734] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00774] [SWS_DM_00775] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00808] [SWS_DM_01366]

▽

△

Requirement	Description	Satisfied by
[RS_AP_00139]	Return type of synchronous function calls	[SWS_DM_00543] [SWS_DM_00594] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00619] [SWS_DM_00638] [SWS_DM_00649] [SWS_DM_00650] [SWS_DM_00651] [SWS_DM_00652] [SWS_DM_00653] [SWS_DM_00654] [SWS_DM_00655] [SWS_DM_00656] [SWS_DM_00666] [SWS_DM_00667] [SWS_DM_00668] [SWS_DM_00669] [SWS_DM_00670] [SWS_DM_00671] [SWS_DM_00672] [SWS_DM_00673] [SWS_DM_00674] [SWS_DM_00692] [SWS_DM_00694] [SWS_DM_00695] [SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00700] [SWS_DM_00714] [SWS_DM_00715] [SWS_DM_00725] [SWS_DM_00735] [SWS_DM_00744] [SWS_DM_00745] [SWS_DM_00755] [SWS_DM_00766] [SWS_DM_00776] [SWS_DM_00792] [SWS_DM_00802] [SWS_DM_00809] [SWS_DM_00918] [SWS_DM_00919] [SWS_DM_01016] [SWS_DM_01088] [SWS_DM_01102] [SWS_DM_01367] [SWS_DM_01698] [SWS_DM_01699] [SWS_DM_01707] [SWS_DM_01708] [SWS_DM_01709] [SWS_DM_01710] [SWS_DM_01711] [SWS_DM_01728] [SWS_DM_01729] [SWS_DM_02076] [SWS_DM_02078]
[RS_AP_00145]	Availability of special member functions	[SWS_DM_00610] [SWS_DM_00611] [SWS_DM_00612] [SWS_DM_00613] [SWS_DM_00973] [SWS_DM_00974] [SWS_DM_00975] [SWS_DM_00976] [SWS_DM_00980]
[RS_AP_00146]	Classes whose construction requires interaction by the ARA framework	[SWS_DM_00609] [SWS_DM_00972]
[RS_AP_00147]	Classes that are created with an InstanceSpecifier as an argument are not copyable, but at most movable.	[SWS_DM_01596] [SWS_DM_01597] [SWS_DM_01598] [SWS_DM_01599] [SWS_DM_01607] [SWS_DM_01608] [SWS_DM_01609] [SWS_DM_01610] [SWS_DM_01611] [SWS_DM_01612] [SWS_DM_01613] [SWS_DM_01614] [SWS_DM_01615] [SWS_DM_01616] [SWS_DM_01617] [SWS_DM_01618] [SWS_DM_01619] [SWS_DM_01620] [SWS_DM_01621] [SWS_DM_01622] [SWS_DM_01623] [SWS_DM_01624] [SWS_DM_01625] [SWS_DM_01626] [SWS_DM_01627] [SWS_DM_01628] [SWS_DM_01629] [SWS_DM_01630] [SWS_DM_01631] [SWS_DM_01632] [SWS_DM_01633] [SWS_DM_01634] [SWS_DM_01635] [SWS_DM_01636] [SWS_DM_01637] [SWS_DM_01638] [SWS_DM_01639] [SWS_DM_01640] [SWS_DM_01641] [SWS_DM_01642] [SWS_DM_01643] [SWS_DM_01644] [SWS_DM_01645] [SWS_DM_01646] [SWS_DM_01647] [SWS_DM_01648] [SWS_DM_01649] [SWS_DM_01650] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_DM_01651] [SWS_DM_01652] [SWS_DM_01653] [SWS_DM_01654] [SWS_DM_01655] [SWS_DM_01656] [SWS_DM_01657] [SWS_DM_01658] [SWS_DM_01659] [SWS_DM_01660] [SWS_DM_01661] [SWS_DM_01662] [SWS_DM_01663] [SWS_DM_01664] [SWS_DM_01665] [SWS_DM_01666] [SWS_DM_01667] [SWS_DM_01668] [SWS_DM_01669] [SWS_DM_01670] [SWS_DM_01671] [SWS_DM_01672] [SWS_DM_01673] [SWS_DM_01674] [SWS_DM_01675] [SWS_DM_01676] [SWS_DM_01677] [SWS_DM_01678] [SWS_DM_01679] [SWS_DM_01680] [SWS_DM_01681] [SWS_DM_01682] [SWS_DM_01683] [SWS_DM_01684] [SWS_DM_01685] [SWS_DM_01686] [SWS_DM_01687] [SWS_DM_01688] [SWS_DM_01689] [SWS_DM_01690] [SWS_DM_01963] [SWS_DM_01964] [SWS_DM_01965] [SWS_DM_01966] [SWS_DM_02063] [SWS_DM_02064] [SWS_DM_02065] [SWS_DM_02066] [SWS_DM_02067] [SWS_DM_02068] [SWS_DM_02069] [SWS_DM_02070] [SWS_DM_02071] [SWS_DM_02072] [SWS_DM_02073] [SWS_DM_02074]
[RS_AP_00158]	IAM access violations	[SWS_DM_01529]
[RS_Diag_00025]	Valid DoIP messages shall be recognized	[SWS_DM_00475] [SWS_DM_01986]
[RS_Diag_00026]	DoIP Vehicle Identification shall be provided	[SWS_DM_00449] [SWS_DM_00720] [SWS_DM_00721] [SWS_DM_00722] [SWS_DM_00723] [SWS_DM_00724] [SWS_DM_00725] [SWS_DM_00726] [SWS_DM_00813] [SWS_DM_00814] [SWS_DM_00855] [SWS_DM_01361] [SWS_DM_01362] [SWS_DM_01363] [SWS_DM_01364] [SWS_DM_01365] [SWS_DM_01366] [SWS_DM_01367] [SWS_DM_01368] [SWS_DM_01496] [SWS_DM_01527] [SWS_DM_01528] [SWS_DM_01568] [SWS_DM_CONSTR_00206]
[RS_Diag_00027]	DoIP diagnostic message shall have a format	[SWS_DM_00449] [SWS_DM_00475] [SWS_DM_01568]
[RS_Diag_00028]	Multiple DoIP sockets shall be allowed on a single port	[SWS_DM_00475]
[RS_Diag_00080]	DoIP shall implement a mechanism to retrieve diagnostic power mode	[SWS_DM_00449] [SWS_DM_00730] [SWS_DM_00731] [SWS_DM_00732] [SWS_DM_00733] [SWS_DM_00734] [SWS_DM_00735] [SWS_DM_00736] [SWS_DM_00814] [SWS_DM_01525] [SWS_DM_01568]
[RS_Diag_00081]	DoIP shall be able to dynamically maintain connection to different testers	[SWS_DM_00475]
[RS_Diag_00082]	DoIP shall implement a mechanism to retrieve Entity Status	[SWS_DM_00449] [SWS_DM_01568] [SWS_DM_02002]
[RS_Diag_00083]	DoIP shall implement a mechanism to check if diagnostic testers are alive	[SWS_DM_00449] [SWS_DM_00475] [SWS_DM_01568]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_00084]	DoIP shall implement routing activation mechanism	[SWS_DM_00449] [SWS_DM_00475] [SWS_DM_01568]
[RS_Diag_00140]	TLS for diagnostic communication (DoIP) shall support at least one ciphersuite as defined in ISO13400-2.	[SWS_DM_01979]
[RS_Diag_04005]	Manage Security Access level handling	[SWS_DM_00103] [SWS_DM_00236] [SWS_DM_00270] [SWS_DM_00271] [SWS_DM_00272] [SWS_DM_00421] [SWS_DM_00478] [SWS_DM_00479] [SWS_DM_00480] [SWS_DM_00760] [SWS_DM_00761] [SWS_DM_00762] [SWS_DM_00763] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00766] [SWS_DM_00767] [SWS_DM_01259] [SWS_DM_01260] [SWS_DM_01261] [SWS_DM_01758]
[RS_Diag_04006]	Manage session handling	[SWS_DM_00101] [SWS_DM_00102] [SWS_DM_00380] [SWS_DM_00381] [SWS_DM_00382] [SWS_DM_00383] [SWS_DM_00701] [SWS_DM_00812] [SWS_DM_00842] [SWS_DM_01747]
[RS_Diag_04016]	Support "Busy handling" by sending a negative response 0x78	[SWS_DM_00368] [SWS_DM_00369] [SWS_DM_01257]
[RS_Diag_04019]	Provide confirmation after transmit diagnostic responses to the application	[SWS_DM_00268] [SWS_DM_00859]
[RS_Diag_04020]	Suppress responses to diagnostic tool requests	[SWS_DM_00433] [SWS_DM_01258]
[RS_Diag_04033]	Support the upload/download services for reading/writing data in an ECU in an extended and manufacturer specific diagnostic session	[SWS_DM_00128] [SWS_DM_00784] [SWS_DM_00785] [SWS_DM_00786] [SWS_DM_00787] [SWS_DM_00788] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00792] [SWS_DM_00793] [SWS_DM_00794] [SWS_DM_00795] [SWS_DM_00796] [SWS_DM_00797] [SWS_DM_00798] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00802] [SWS_DM_00803] [SWS_DM_00868] [SWS_DM_00869] [SWS_DM_00871] [SWS_DM_01096] [SWS_DM_01097]
[RS_Diag_04058]	Ability to access different event memories	[SWS_DM_00055] [SWS_DM_00056] [SWS_DM_00057]
[RS_Diag_04063]	Process a dedicated event identifier for each monitoring path to support an autonomous handling of different events/faults	[SWS_DM_01701] [SWS_DM_01731] [SWS_DM_01732]
[RS_Diag_04064]	Provide configurable buffer sizes for storage of the events, status information and environmental data	[SWS_DM_00920]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04067]	Provide the diagnostic status information according to ISO 14229-1	[SWS_DM_00061] [SWS_DM_00062] [SWS_DM_00063] [SWS_DM_00213] [SWS_DM_00217] [SWS_DM_00218] [SWS_DM_00244] [SWS_DM_00245] [SWS_DM_00246] [SWS_DM_00370] [SWS_DM_00371] [SWS_DM_00372] [SWS_DM_00373] [SWS_DM_00374] [SWS_DM_00658] [SWS_DM_00659] [SWS_DM_00811] [SWS_DM_00883] [SWS_DM_00932] [SWS_DM_00933] [SWS_DM_00945] [SWS_DM_00946] [SWS_DM_00966] [SWS_DM_00967] [SWS_DM_00968] [SWS_DM_01037] [SWS_DM_01114] [SWS_DM_01115] [SWS_DM_01256] [SWS_DM_01569]
[RS_Diag_04068]	The diagnostic in AUTOSAR shall support event specific debounce counters to improve signal quality internally (According to ISO 14229-1 Appendix D)	[SWS_DM_00014] [SWS_DM_00018] [SWS_DM_00021] [SWS_DM_00022] [SWS_DM_00023] [SWS_DM_00024] [SWS_DM_00025] [SWS_DM_00029] [SWS_DM_00039] [SWS_DM_00086] [SWS_DM_00538] [SWS_DM_00549] [SWS_DM_00621] [SWS_DM_00622] [SWS_DM_00623] [SWS_DM_00624] [SWS_DM_00625] [SWS_DM_00626] [SWS_DM_00627] [SWS_DM_00628] [SWS_DM_00645] [SWS_DM_00654] [SWS_DM_00656] [SWS_DM_00875] [SWS_DM_00876] [SWS_DM_00880] [SWS_DM_00952] [SWS_DM_01099] [SWS_DM_01101] [SWS_DM_01268] [SWS_DM_01583] [SWS_DM_01584] [SWS_DM_01702] [SWS_DM_01710]
[RS_Diag_04071]	Process events according to their defined importance like priority and/or severity	[SWS_DM_00916] [SWS_DM_CONSTR_00961]
[RS_Diag_04091]	Application access to snapshot record data	[SWS_DM_00660] [SWS_DM_00661] [SWS_DM_00662] [SWS_DM_00668] [SWS_DM_00894]
[RS_Diag_04093]	Memory overflow indication	[SWS_DM_00918] [SWS_DM_00919] [SWS_DM_00921] [SWS_DM_00922] [SWS_DM_00923] [SWS_DM_00924] [SWS_DM_00925] [SWS_DM_00926] [SWS_DM_01354] [SWS_DM_02076]
[RS_Diag_04097]	Decentralized and modular diagnostic configuration in applications	[SWS_DM_00393] [SWS_DM_00401] [SWS_DM_00570] [SWS_DM_00572] [SWS_DM_00848] [SWS_DM_00849] [SWS_DM_00905] [SWS_DM_00906] [SWS_DM_00908] [SWS_DM_01038] [SWS_DM_01039] [SWS_DM_01565] [SWS_DM_CONSTR_00394] [SWS_DM_CONSTR_00395] [SWS_DM_CONSTR_00396]
[RS_Diag_04105]	Event memory management	[SWS_DM_00148] [SWS_DM_00150] [SWS_DM_00657] [SWS_DM_00664] [SWS_DM_00928] [SWS_DM_00929] [SWS_DM_00947] [SWS_DM_01579] [SWS_DM_01734]
[RS_Diag_04109]	Provide an interface to retrieve the number of event memory entries	[SWS_DM_00669] [SWS_DM_00670] [SWS_DM_00902] [SWS_DM_01352]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04115]	The optional parameter DTCSetting ControlOptionRecord as part of UDS service ControlDTCSetting shall be limited to GroupOfDTC	[SWS_DM_00064] [SWS_DM_00231]
[RS_Diag_04117]	Configurable behavior for DTC deletion	[SWS_DM_00064] [SWS_DM_00065] [SWS_DM_00091] [SWS_DM_00092] [SWS_DM_00116] [SWS_DM_00117] [SWS_DM_00121] [SWS_DM_00122] [SWS_DM_00123] [SWS_DM_00124] [SWS_DM_00144] [SWS_DM_00145] [SWS_DM_00146] [SWS_DM_00147] [SWS_DM_00159] [SWS_DM_00160] [SWS_DM_00896] [SWS_DM_01578] [SWS_DM_CONSTR_00082]
[RS_Diag_04118]	Optionally support event displacement	[SWS_DM_00916] [SWS_DM_00927] [SWS_DM_00928] [SWS_DM_00929] [SWS_DM_00930] [SWS_DM_00932] [SWS_DM_00933] [SWS_DM_00934] [SWS_DM_01569] [SWS_DM_CONSTR_00961]
[RS_Diag_04120]	Support a predefined AddressAnd LengthFormatIdentifier	[SWS_DM_00129] [SWS_DM_00130]
[RS_Diag_04125]	Event debounce counter shall be configurable	[SWS_DM_00017] [SWS_DM_00021] [SWS_DM_00024] [SWS_DM_00025] [SWS_DM_00029] [SWS_DM_00378] [SWS_DM_00875] [SWS_DM_00876] [SWS_DM_00882] [SWS_DM_00948] [SWS_DM_01094] [SWS_DM_01095] [SWS_DM_01349]
[RS_Diag_04127]	Configurable record numbers and trigger options for DTCSnapshot Records and DTCExtendedData Records	[SWS_DM_00895] [SWS_DM_00949] [SWS_DM_00953] [SWS_DM_01085] [SWS_DM_01086] [SWS_DM_01267] [SWS_DM_01276] [SWS_DM_01277]
[RS_Diag_04131]	Consistent event management mechanisms	[SWS_DM_01960]
[RS_Diag_04133]	Aging for event memory entries	[SWS_DM_00237] [SWS_DM_00238] [SWS_DM_00239] [SWS_DM_00240] [SWS_DM_00241] [SWS_DM_01264] [SWS_DM_01265] [SWS_DM_01576] [SWS_DM_01953] [SWS_DM_CONSTR_00960]
[RS_Diag_04135]	Support UDS service $38 (Request FileTransfer)	[SWS_DM_01310] [SWS_DM_01311] [SWS_DM_01312] [SWS_DM_01313] [SWS_DM_01314] [SWS_DM_01315] [SWS_DM_01316] [SWS_DM_01317] [SWS_DM_01318] [SWS_DM_01319] [SWS_DM_01320] [SWS_DM_01321] [SWS_DM_01322] [SWS_DM_01323] [SWS_DM_01324] [SWS_DM_01325] [SWS_DM_01326] [SWS_DM_01327] [SWS_DM_01328] [SWS_DM_01329] [SWS_DM_01330] [SWS_DM_01331] [SWS_DM_01332] [SWS_DM_01333] [SWS_DM_01334] [SWS_DM_01335] [SWS_DM_01336] [SWS_DM_01337] [SWS_DM_01339] [SWS_DM_01340] [SWS_DM_01497] [SWS_DM_01498] [SWS_DM_01499] [SWS_DM_01500] [SWS_DM_01501] [SWS_DM_01502] [SWS_DM_01503] [SWS_DM_01504] [SWS_DM_01505] [SWS_DM_01506] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_DM_01507] [SWS_DM_01508] [SWS_DM_01509] [SWS_DM_01510] [SWS_DM_01511] [SWS_DM_01512] [SWS_DM_01513] [SWS_DM_01514] [SWS_DM_01515] [SWS_DM_01516] [SWS_DM_01517] [SWS_DM_01518] [SWS_DM_01519] [SWS_DM_01520] [SWS_DM_01521] [SWS_DM_01522] [SWS_DM_01523] [SWS_DM_01530] [SWS_DM_01531] [SWS_DM_01532] [SWS_DM_01533] [SWS_DM_01534] [SWS_DM_01535] [SWS_DM_01536] [SWS_DM_01537] [SWS_DM_01538] [SWS_DM_01539] [SWS_DM_01540] [SWS_DM_01541] [SWS_DM_01542] [SWS_DM_01543] [SWS_DM_01544] [SWS_DM_01545] [SWS_DM_01546] [SWS_DM_01547] [SWS_DM_01548] [SWS_DM_01549] [SWS_DM_01550] [SWS_DM_01551] [SWS_DM_01552] [SWS_DM_01553] [SWS_DM_01554] [SWS_DM_01555] [SWS_DM_01556] [SWS_DM_01557] [SWS_DM_01558] [SWS_DM_01559] [SWS_DM_01560] [SWS_DM_02079]
[RS_Diag_04136]	Configurable "confirmed" threshold	[SWS_DM_00218]
[RS_Diag_04140]	Aging for UDS status bits "confirmed DTC" and "testFailedSinceLastClear"	[SWS_DM_00243] [SWS_DM_01577]
[RS_Diag_04147]	Communication with the transport layers to receive and send diagnostic data	[SWS_DM_00291] [SWS_DM_00293] [SWS_DM_00294] [SWS_DM_00296] [SWS_DM_00297] [SWS_DM_00298] [SWS_DM_00299] [SWS_DM_00300] [SWS_DM_00301] [SWS_DM_00302] [SWS_DM_00303] [SWS_DM_00304] [SWS_DM_00306] [SWS_DM_00307] [SWS_DM_00309] [SWS_DM_00310] [SWS_DM_00311] [SWS_DM_00312] [SWS_DM_00313] [SWS_DM_00314] [SWS_DM_00315] [SWS_DM_00319] [SWS_DM_00322] [SWS_DM_00323] [SWS_DM_00325] [SWS_DM_00326] [SWS_DM_00327] [SWS_DM_00336] [SWS_DM_00337] [SWS_DM_00338] [SWS_DM_00384] [SWS_DM_09010] [SWS_DM_09012] [SWS_DM_09015] [SWS_DM_09016] [SWS_DM_09017] [SWS_DM_09021] [SWS_DM_09025]
[RS_Diag_04148]	Provide capabilities to inform applications about diagnostic data changes	[SWS_DM_00667] [SWS_DM_00894]
[RS_Diag_04150]	Support the primary fault memory defined by ISO 14229-1	[SWS_DM_00055] [SWS_DM_00056] [SWS_DM_00083] [SWS_DM_00657] [SWS_DM_00664] [SWS_DM_00911] [SWS_DM_CONSTR_00084]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04151]	Event status handling	[SWS_DM_00218] [SWS_DM_00643] [SWS_DM_00644] [SWS_DM_00646] [SWS_DM_00647] [SWS_DM_00648] [SWS_DM_00649] [SWS_DM_00652] [SWS_DM_00655] [SWS_DM_00658] [SWS_DM_00659] [SWS_DM_01024] [SWS_DM_01025] [SWS_DM_01026] [SWS_DM_01703] [SWS_DM_01704] [SWS_DM_01705] [SWS_DM_01706] [SWS_DM_01707] [SWS_DM_01749] [SWS_DM_01750] [SWS_DM_01751] [SWS_DM_01752] [SWS_DM_01753] [SWS_DM_01754] [SWS_DM_01755] [SWS_DM_01756] [SWS_DM_01757]
[RS_Diag_04157]	Reporting of DTCs and related data	[SWS_DM_00058] [SWS_DM_00061] [SWS_DM_00062] [SWS_DM_00063] [SWS_DM_00244] [SWS_DM_00245] [SWS_DM_00246] [SWS_DM_00247] [SWS_DM_00370] [SWS_DM_00371] [SWS_DM_00372] [SWS_DM_00373] [SWS_DM_00374] [SWS_DM_00966] [SWS_DM_00967] [SWS_DM_00968] [SWS_DM_01256]
[RS_Diag_04159]	Control of DTC storage	[SWS_DM_00229] [SWS_DM_00378] [SWS_DM_00663] [SWS_DM_00672] [SWS_DM_00673] [SWS_DM_00674] [SWS_DM_00811] [SWS_DM_00909] [SWS_DM_00910] [SWS_DM_01094] [SWS_DM_01095] [SWS_DM_01353]
[RS_Diag_04160]	ResponseOnEvent according to ISO 14229-1	[SWS_DM_00491] [SWS_DM_00493] [SWS_DM_00494] [SWS_DM_01098] [SWS_DM_01117] [SWS_DM_01118] [SWS_DM_01119] [SWS_DM_01120] [SWS_DM_01121] [SWS_DM_01122] [SWS_DM_01262] [SWS_DM_01263]
[RS_Diag_04165]	Triggering of multiple events upon a master event is reported	[SWS_DM_01733]
[RS_Diag_04166]	Several tester conversations in parallel with assigned priorities	[SWS_DM_00426] [SWS_DM_00428] [SWS_DM_00429] [SWS_DM_00430] [SWS_DM_00690] [SWS_DM_00691] [SWS_DM_00693] [SWS_DM_00701] [SWS_DM_00782] [SWS_DM_00783] [SWS_DM_00840] [SWS_DM_00841] [SWS_DM_00843] [SWS_DM_00844] [SWS_DM_00856] [SWS_DM_00935] [SWS_DM_00936] [SWS_DM_00937] [SWS_DM_00938] [SWS_DM_00939] [SWS_DM_00940] [SWS_DM_00941] [SWS_DM_00942] [SWS_DM_00943] [SWS_DM_00944] [SWS_DM_01581] [SWS_DM_01980] [SWS_DM_01981] [SWS_DM_01982] [SWS_DM_01983] [SWS_DM_01984] [SWS_DM_01985] [SWS_DM_02003] [SWS_DM_02004]
[RS_Diag_04167]	Conversation preemption/abortion	[SWS_DM_00049] [SWS_DM_00277] [SWS_DM_00278] [SWS_DM_00279] [SWS_DM_00280] [SWS_DM_00290] [SWS_DM_00431] [SWS_DM_00577] [SWS_DM_00847] [SWS_DM_00984] [SWS_DM_01348]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04168]	Adding of user-defined transport layers	[SWS_DM_00306] [SWS_DM_00307] [SWS_DM_00309] [SWS_DM_00310] [SWS_DM_00311] [SWS_DM_00312] [SWS_DM_00313] [SWS_DM_00314] [SWS_DM_00315] [SWS_DM_00319] [SWS_DM_00322] [SWS_DM_00323] [SWS_DM_00325] [SWS_DM_00326] [SWS_DM_00327] [SWS_DM_00329] [SWS_DM_00330] [SWS_DM_00331] [SWS_DM_00332] [SWS_DM_00333] [SWS_DM_00336] [SWS_DM_00340] [SWS_DM_00342] [SWS_DM_00345] [SWS_DM_00346] [SWS_DM_00347] [SWS_DM_00348] [SWS_DM_00349] [SWS_DM_00350] [SWS_DM_00351] [SWS_DM_00356] [SWS_DM_00357] [SWS_DM_00358] [SWS_DM_00359] [SWS_DM_00384] [SWS_DM_00385] [SWS_DM_00386] [SWS_DM_00387] [SWS_DM_00388] [SWS_DM_00389] [SWS_DM_00392] [SWS_DM_00487] [SWS_DM_01741] [SWS_DM_01742] [SWS_DM_01743] [SWS_DM_01744] [SWS_DM_01745] [SWS_DM_01746] [SWS_DM_09015] [SWS_DM_09016] [SWS_DM_09017] [SWS_DM_09021] [SWS_DM_09025]
[RS_Diag_04169]	Provide an interface for external UDS service processors.	[SWS_DM_00551] [SWS_DM_00552] [SWS_DM_00553] [SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00556] [SWS_DM_00557] [SWS_DM_00558] [SWS_DM_00578] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00584] [SWS_DM_00586] [SWS_DM_00588] [SWS_DM_00590] [SWS_DM_00594] [SWS_DM_00595] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00600] [SWS_DM_00601] [SWS_DM_00602] [SWS_DM_00603] [SWS_DM_00604] [SWS_DM_00605] [SWS_DM_00607] [SWS_DM_00608] [SWS_DM_00609] [SWS_DM_00610] [SWS_DM_00611] [SWS_DM_00612] [SWS_DM_00613] [SWS_DM_00614] [SWS_DM_00615] [SWS_DM_00616] [SWS_DM_00617] [SWS_DM_00618] [SWS_DM_00619] [SWS_DM_00620] [SWS_DM_00631] [SWS_DM_00632] [SWS_DM_00633] [SWS_DM_00634] [SWS_DM_00635] [SWS_DM_00636] [SWS_DM_00637] [SWS_DM_00638] [SWS_DM_00639] [SWS_DM_00641] [SWS_DM_00690] [SWS_DM_00691] [SWS_DM_00692] [SWS_DM_00693] [SWS_DM_00694] [SWS_DM_00695] [SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00700] [SWS_DM_00701] [SWS_DM_00707] [SWS_DM_00708] [SWS_DM_00782] [SWS_DM_00783] [SWS_DM_00865] [SWS_DM_01007] [SWS_DM_01009] [SWS_DM_01010] [SWS_DM_01011] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_DM_01012] [SWS_DM_01013] [SWS_DM_01014] [SWS_DM_01017] [SWS_DM_01088] [SWS_DM_01089] [SWS_DM_01355] [SWS_DM_01356] [SWS_DM_01357] [SWS_DM_01699] [SWS_DM_01700] [SWS_DM_02075] [SWS_DM_80003] [SWS_DM_80004] [SWS_DM_80005] [SWS_DM_80013] [SWS_DM_80023] [SWS_DM_CONSTR_00397]
[RS_Diag_04170]	Provide connection specific meta information to external service processors	[SWS_DM_00294] [SWS_DM_00302] [SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00556] [SWS_DM_00618] [SWS_DM_00636] [SWS_DM_00637] [SWS_DM_00692] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00774] [SWS_DM_00775] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00808] [SWS_DM_00971] [SWS_DM_00972] [SWS_DM_00973] [SWS_DM_00974] [SWS_DM_00975] [SWS_DM_00976] [SWS_DM_00977] [SWS_DM_00978] [SWS_DM_00979] [SWS_DM_00980] [SWS_DM_01342] [SWS_DM_01343] [SWS_DM_01344] [SWS_DM_01345] [SWS_DM_01818]
[RS_Diag_04172]	Inform external service processors about outcome of the final response	[SWS_DM_00307] [SWS_DM_00312] [SWS_DM_00578] [SWS_DM_00618] [SWS_DM_00632] [SWS_DM_00636] [SWS_DM_00641] [SWS_DM_00859]
[RS_Diag_04173]	Different signature types, when delegating processing of UDS service to the application	[SWS_DM_00618] [SWS_DM_01759]
[RS_Diag_04174]	Provide SA and TA to external service processors	[SWS_DM_00297] [SWS_DM_00298] [SWS_DM_00299]
[RS_Diag_04177]	Custom diagnostic services	[SWS_DM_00502] [SWS_DM_00983]
[RS_Diag_04178]	Support operation cycles according to ISO 14229-1	[SWS_DM_00751] [SWS_DM_00752] [SWS_DM_00753] [SWS_DM_00755] [SWS_DM_01027] [SWS_DM_01102] [SWS_DM_01103] [SWS_DM_01104] [SWS_DM_01105] [SWS_DM_01358] [SWS_DM_02078] [SWS_DM_CONSTR_00168]
[RS_Diag_04179]	Provide interfaces for monitoring application.	[SWS_DM_00540] [SWS_DM_00541] [SWS_DM_00542] [SWS_DM_00543] [SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00965] [SWS_DM_01694] [SWS_DM_01695] [SWS_DM_01696] [SWS_DM_01697] [SWS_DM_01698] [SWS_DM_01702] [SWS_DM_01732] [SWS_DM_01972] [SWS_DM_01973]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04180]	Process all UDS Services related to diagnostic fault memory of ISO 14229-1 internally	[SWS_DM_00062] [SWS_DM_00090] [SWS_DM_00091] [SWS_DM_00092] [SWS_DM_00115] [SWS_DM_00162] [SWS_DM_00163] [SWS_DM_00164] [SWS_DM_00217] [SWS_DM_00229] [SWS_DM_00244] [SWS_DM_00245] [SWS_DM_00246] [SWS_DM_00247] [SWS_DM_00370] [SWS_DM_00371] [SWS_DM_00372] [SWS_DM_00373] [SWS_DM_00374] [SWS_DM_00909] [SWS_DM_00910] [SWS_DM_00966] [SWS_DM_00967] [SWS_DM_00968] [SWS_DM_01028] [SWS_DM_01256]
[RS_Diag_04182]	Provide an application interface to change operation cycles states	[SWS_DM_01027] [SWS_DM_01102] [SWS_DM_01103]
[RS_Diag_04183]	Notify interested parties about event status changes	[SWS_DM_00650] [SWS_DM_00886] [SWS_DM_01029] [SWS_DM_01030] [SWS_DM_01031] [SWS_DM_01350] [SWS_DM_01351] [SWS_DM_01708]
[RS_Diag_04185]	Notify applications about the clearing of an event	[SWS_DM_00562]
[RS_Diag_04186]	Notify applications about the start or restart of an operation cycle	[SWS_DM_00563] [SWS_DM_00755] [SWS_DM_01358] [SWS_DM_02078]
[RS_Diag_04189]	Support a fine grained configuration for SnapshotRecords and Extended DataRecords	[SWS_DM_00151] [SWS_DM_00155]
[RS_Diag_04190]	Usage of internal data elements in SnapshotRecords and ExtendedData Records	[SWS_DM_00017] [SWS_DM_00030] [SWS_DM_00152] [SWS_DM_00154] [SWS_DM_00921] [SWS_DM_00949] [SWS_DM_00950] [SWS_DM_00951] [SWS_DM_00952] [SWS_DM_00953] [SWS_DM_00954] [SWS_DM_00955] [SWS_DM_00956] [SWS_DM_00957] [SWS_DM_00958] [SWS_DM_00959] [SWS_DM_00961] [SWS_DM_00962] [SWS_DM_00963] [SWS_DM_00964] [SWS_DM_01954] [SWS_DM_01955] [SWS_DM_01956] [SWS_DM_01957] [SWS_DM_01958] [SWS_DM_01959]
[RS_Diag_04192]	Provide the ability to handle event specific enable conditions	[SWS_DM_00568] [SWS_DM_00710] [SWS_DM_00711] [SWS_DM_00712] [SWS_DM_00713] [SWS_DM_00714] [SWS_DM_00715] [SWS_DM_00882] [SWS_DM_01093] [SWS_DM_01095] [SWS_DM_01726] [SWS_DM_01727] [SWS_DM_01728] [SWS_DM_01729] [SWS_DM_01730] [SWS_DM_01735] [SWS_DM_01736] [SWS_DM_01737]
[RS_Diag_04194]	ClearDTC shall be accessible for applications	[SWS_DM_00671] [SWS_DM_00897] [SWS_DM_00898] [SWS_DM_00899] [SWS_DM_00900] [SWS_DM_00901]
[RS_Diag_04195]	Chronological reporting order of the DTCs located in the configured event memory	[SWS_DM_00981] [SWS_DM_00982] [SWS_DM_01566]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04196]	UDS Service handling for all diagnostic services defined in ISO 14229-2	[SWS_DM_00090] [SWS_DM_00096] [SWS_DM_00097] [SWS_DM_00113] [SWS_DM_00126] [SWS_DM_00127] [SWS_DM_00128] [SWS_DM_00134] [SWS_DM_00137] [SWS_DM_00140] [SWS_DM_00141] [SWS_DM_00162] [SWS_DM_00170] [SWS_DM_00186] [SWS_DM_00199] [SWS_DM_00201] [SWS_DM_00227] [SWS_DM_00234] [SWS_DM_00235] [SWS_DM_00236] [SWS_DM_00269] [SWS_DM_00360] [SWS_DM_00363] [SWS_DM_00574] [SWS_DM_00575] [SWS_DM_00576] [SWS_DM_00784] [SWS_DM_00785] [SWS_DM_00786] [SWS_DM_00787] [SWS_DM_00788] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00792] [SWS_DM_00793] [SWS_DM_00794] [SWS_DM_00795] [SWS_DM_00796] [SWS_DM_00797] [SWS_DM_00798] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00802] [SWS_DM_00803] [SWS_DM_00804] [SWS_DM_00805] [SWS_DM_00806] [SWS_DM_00807] [SWS_DM_00808] [SWS_DM_00809] [SWS_DM_00810] [SWS_DM_00860] [SWS_DM_00866] [SWS_DM_00867] [SWS_DM_01007] [SWS_DM_01009] [SWS_DM_01010] [SWS_DM_01011] [SWS_DM_01012] [SWS_DM_01013] [SWS_DM_01014] [SWS_DM_01017] [SWS_DM_01020] [SWS_DM_01021] [SWS_DM_01022] [SWS_DM_01023] [SWS_DM_01090] [SWS_DM_01092] [SWS_DM_01258] [SWS_DM_01270] [SWS_DM_01271] [SWS_DM_01272] [SWS_DM_01309] [SWS_DM_01346] [SWS_DM_01347] [SWS_DM_01562] [SWS_DM_01563] [SWS_DM_01564] [SWS_DM_02059] [SWS_DM_02060]
[RS_Diag_04197]	Clearing the user defined fault memory	[SWS_DM_00193] [SWS_DM_00194] [SWS_DM_00195] [SWS_DM_00208]
[RS_Diag_04198]	Process all UDS Services related to session and security management of ISO 14229 internally	[SWS_DM_00226] [SWS_DM_00228]
[RS_Diag_04199]	UDS service request validation	[SWS_DM_00111] [SWS_DM_00112] [SWS_DM_00286] [SWS_DM_00287] [SWS_DM_00288] [SWS_DM_00289] [SWS_DM_00770] [SWS_DM_00771] [SWS_DM_00772] [SWS_DM_00773] [SWS_DM_00774] [SWS_DM_00775] [SWS_DM_00776] [SWS_DM_00777] [SWS_DM_00860] [SWS_DM_00970] [SWS_DM_01252] [SWS_DM_01253] [SWS_DM_01254] [SWS_DM_01255]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04200]	Support event combination	[SWS_DM_01106] [SWS_DM_01107] [SWS_DM_01108] [SWS_DM_01109] [SWS_DM_01110] [SWS_DM_01111] [SWS_DM_01112] [SWS_DM_01113] [SWS_DM_01114] [SWS_DM_01115] [SWS_DM_01116] [SWS_DM_01269]
[RS_Diag_04201]	Support a configuration to assign specific events to a customer specific DTC	[SWS_DM_00060] [SWS_DM_00642] [SWS_DM_00653] [SWS_DM_01709] [SWS_DM_01733] [SWS_DM_CONSTR_00059]
[RS_Diag_04203]	Common checks on all supported UDS Services Requests	[SWS_DM_00096] [SWS_DM_00098] [SWS_DM_00099] [SWS_DM_00100] [SWS_DM_00101] [SWS_DM_00102] [SWS_DM_00103] [SWS_DM_00202] [SWS_DM_00203] [SWS_DM_00231] [SWS_DM_00252] [SWS_DM_00409] [SWS_DM_00412] [SWS_DM_00413] [SWS_DM_00414] [SWS_DM_00415] [SWS_DM_00416] [SWS_DM_00417] [SWS_DM_00437] [SWS_DM_00448] [SWS_DM_00450] [SWS_DM_00507] [SWS_DM_00863] [SWS_DM_00864]
[RS_Diag_04204]	Provide the current status of each warning indicator.	[SWS_DM_00223] [SWS_DM_00224] [SWS_DM_00651] [SWS_DM_00740] [SWS_DM_00741] [SWS_DM_00742] [SWS_DM_00743] [SWS_DM_00744] [SWS_DM_00745] [SWS_DM_00888] [SWS_DM_01032] [SWS_DM_01033] [SWS_DM_01034] [SWS_DM_01035] [SWS_DM_01266] [SWS_DM_01359] [SWS_DM_01582] [SWS_DM_01974] [SWS_DM_01975]
[RS_Diag_04205]	Support of SnapshotRecords	[SWS_DM_00151] [SWS_DM_00152] [SWS_DM_00660] [SWS_DM_00661] [SWS_DM_00662] [SWS_DM_00668] [SWS_DM_00969] [SWS_DM_01085] [SWS_DM_01086] [SWS_DM_01087] [SWS_DM_01276] [SWS_DM_01277]
[RS_Diag_04206]	Support of ExtendedDataRecords	[SWS_DM_00154] [SWS_DM_00155] [SWS_DM_00895]
[RS_Diag_04208]	Inform the application about diagnostic session and diagnostic security level changes on each tester connection.	[SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00707] [SWS_DM_00708] [SWS_DM_00845] [SWS_DM_00846] [SWS_DM_01356] [SWS_DM_01357]
[RS_Diag_04209]	Pseudo parallel client interaction according to ISO	[SWS_DM_00690] [SWS_DM_00691] [SWS_DM_00701] [SWS_DM_00782] [SWS_DM_00783]
[RS_Diag_04211]	Persistent storage of DTC status and environmental data	[SWS_DM_00148] [SWS_DM_00150] [SWS_DM_00811] [SWS_DM_01579]
[RS_Diag_04214]	Support the user defined fault memories defined by ISO 14229-1	[SWS_DM_00055] [SWS_DM_00057] [SWS_DM_00083] [SWS_DM_00911] [SWS_DM_CONSTR_00084]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04215]	Support of UDS service ReadDataBy PeriodicIdentifier (0x2A)	[SWS_DM_01040] [SWS_DM_01041] [SWS_DM_01042] [SWS_DM_01043] [SWS_DM_01044] [SWS_DM_01045] [SWS_DM_01046] [SWS_DM_01047] [SWS_DM_01048] [SWS_DM_01049] [SWS_DM_01050] [SWS_DM_01051] [SWS_DM_01052] [SWS_DM_01053] [SWS_DM_01054] [SWS_DM_01055] [SWS_DM_01056] [SWS_DM_01057] [SWS_DM_01058] [SWS_DM_01059] [SWS_DM_01060] [SWS_DM_01061] [SWS_DM_01062] [SWS_DM_01063] [SWS_DM_01064] [SWS_DM_01065] [SWS_DM_01066] [SWS_DM_01067] [SWS_DM_01068] [SWS_DM_01069] [SWS_DM_02089]
[RS_Diag_04216]	Support for multiple Diagnostic Server Instances	[SWS_DM_00390] [SWS_DM_00391] [SWS_DM_00420]
[RS_Diag_04224]	Support the UDS service 0x31 (RoutineControl) according to ISO 14229-1	[SWS_DM_00201] [SWS_DM_00202] [SWS_DM_00203] [SWS_DM_00437] [SWS_DM_00448] [SWS_DM_00551] [SWS_DM_00552] [SWS_DM_00553] [SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00556] [SWS_DM_00557] [SWS_DM_00558] [SWS_DM_00574] [SWS_DM_00575] [SWS_DM_00576] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00594] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00605] [SWS_DM_00633] [SWS_DM_01270] [SWS_DM_01271] [SWS_DM_01272] [SWS_DM_80003] [SWS_DM_80004] [SWS_DM_80005] [SWS_DM_80013] [SWS_DM_80023]
[RS_Diag_04225]	The diagnostic in AUTOSAR shall support event specific time base debounce counters	[SWS_DM_00015] [SWS_DM_00030] [SWS_DM_00033] [SWS_DM_00036] [SWS_DM_00038] [SWS_DM_00039] [SWS_DM_00085] [SWS_DM_00086] [SWS_DM_00539] [SWS_DM_00550] [SWS_DM_00629] [SWS_DM_00630] [SWS_DM_00645] [SWS_DM_00654] [SWS_DM_00877] [SWS_DM_00878] [SWS_DM_00880] [SWS_DM_01583] [SWS_DM_01584] [SWS_DM_01697] [SWS_DM_01710]
[RS_Diag_04226]	Diagnostic session check	[SWS_DM_00101] [SWS_DM_00102] [SWS_DM_00413] [SWS_DM_00416] [SWS_DM_00448] [SWS_DM_01046] [SWS_DM_01056] [SWS_DM_01079] [SWS_DM_01081] [SWS_DM_01261] [SWS_DM_01951]
[RS_Diag_04227]	Common check for Security Access	[SWS_DM_00103] [SWS_DM_00414] [SWS_DM_00417] [SWS_DM_00437] [SWS_DM_00450] [SWS_DM_01047] [SWS_DM_01056] [SWS_DM_01080] [SWS_DM_01082] [SWS_DM_01260] [SWS_DM_01952]
[RS_Diag_04228]	Common check for Message lengths	[SWS_DM_00098] [SWS_DM_00412] [SWS_DM_00507] [SWS_DM_01041] [SWS_DM_01042]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04232]	Access rights in client certificates	[SWS_DM_01224] [SWS_DM_01225]
[RS_Diag_04233]	Access granularity of diagnostic services	[SWS_DM_01223] [SWS_DM_01739]
[RS_Diag_04234]	Binary compatibility of white list for individual access	[SWS_DM_01218] [SWS_DM_01219] [SWS_DM_01220] [SWS_DM_01221] [SWS_DM_01222]
[RS_Diag_04239]	Diagnostic services in deauthenticated state	[SWS_DM_01209]
[RS_Diag_04240]	Application based authentication	[SWS_DM_01202] [SWS_DM_01203] [SWS_DM_01204] [SWS_DM_01205] [SWS_DM_01206] [SWS_DM_01207] [SWS_DM_01208] [SWS_DM_01210] [SWS_DM_01211] [SWS_DM_01212] [SWS_DM_01213] [SWS_DM_01214] [SWS_DM_01215] [SWS_DM_01216] [SWS_DM_01217] [SWS_DM_01218] [SWS_DM_01219] [SWS_DM_01220] [SWS_DM_01221] [SWS_DM_01222] [SWS_DM_01570]
[RS_Diag_04242]	The DoIP module shall support Vehicle Internal Testers.	[SWS_DM_00475] [SWS_DM_00815] [SWS_DM_00816] [SWS_DM_00820] [SWS_DM_00821] [SWS_DM_00822] [SWS_DM_00830] [SWS_DM_00831] [SWS_DM_00832] [SWS_DM_00833] [SWS_DM_00834] [SWS_DM_00835] [SWS_DM_00836] [SWS_DM_00837] [SWS_DM_01526]
[RS_Diag_04244]	Support sub-function 0x04 of UDS service 0x19.	[SWS_DM_00246]
[RS_Diag_04245]	Support sub-function 0x06 of UDS service 0x19.	[SWS_DM_00370]
[RS_Diag_04246]	Support of UDS service Dynamically DefineDataIdentifier (0x2C) with subfunction 0x01 (defineByIdentifier)	[SWS_DM_01070] [SWS_DM_01071] [SWS_DM_01072] [SWS_DM_01073] [SWS_DM_01074] [SWS_DM_01075] [SWS_DM_01076] [SWS_DM_01077] [SWS_DM_01078] [SWS_DM_01079] [SWS_DM_01080] [SWS_DM_01081] [SWS_DM_01082] [SWS_DM_01083]
[RS_Diag_04248]	Support of session control service	[SWS_DM_00226] [SWS_DM_00227] [SWS_DM_00228]
[RS_Diag_04249]	Support of session layer service	[SWS_DM_00368] [SWS_DM_00369] [SWS_DM_00380] [SWS_DM_00381] [SWS_DM_00382] [SWS_DM_00383] [SWS_DM_00812] [SWS_DM_01257] [SWS_DM_01258] [SWS_DM_01747]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04251]	Support of UDS service 0x29 (Authentication)	[SWS_DM_01123] [SWS_DM_01124] [SWS_DM_01125] [SWS_DM_01126] [SWS_DM_01127] [SWS_DM_01128] [SWS_DM_01130] [SWS_DM_01131] [SWS_DM_01132] [SWS_DM_01133] [SWS_DM_01134] [SWS_DM_01136] [SWS_DM_01137] [SWS_DM_01138] [SWS_DM_01139] [SWS_DM_01140] [SWS_DM_01141] [SWS_DM_01142] [SWS_DM_01143] [SWS_DM_01144] [SWS_DM_01145] [SWS_DM_01146] [SWS_DM_01147] [SWS_DM_01148] [SWS_DM_01149] [SWS_DM_01150] [SWS_DM_01151] [SWS_DM_01152] [SWS_DM_01153] [SWS_DM_01154] [SWS_DM_01155] [SWS_DM_01156] [SWS_DM_01157] [SWS_DM_01158] [SWS_DM_01159] [SWS_DM_01160] [SWS_DM_01161] [SWS_DM_01162] [SWS_DM_01163] [SWS_DM_01164] [SWS_DM_01165] [SWS_DM_01166] [SWS_DM_01167] [SWS_DM_01168] [SWS_DM_01169] [SWS_DM_01170] [SWS_DM_01171] [SWS_DM_01172] [SWS_DM_01173] [SWS_DM_01174] [SWS_DM_01175] [SWS_DM_01176] [SWS_DM_01177] [SWS_DM_01178] [SWS_DM_01179] [SWS_DM_01180] [SWS_DM_01181] [SWS_DM_01182] [SWS_DM_01183] [SWS_DM_01184] [SWS_DM_01185] [SWS_DM_01186] [SWS_DM_01187] [SWS_DM_01188] [SWS_DM_01189] [SWS_DM_01190] [SWS_DM_01191] [SWS_DM_01192] [SWS_DM_01193] [SWS_DM_01194] [SWS_DM_01195] [SWS_DM_01196] [SWS_DM_01197] [SWS_DM_01198] [SWS_DM_01199] [SWS_DM_01200] [SWS_DM_01201] [SWS_DM_01226] [SWS_DM_01227] [SWS_DM_01228] [SWS_DM_01229] [SWS_DM_01230] [SWS_DM_01231] [SWS_DM_01233] [SWS_DM_01235] [SWS_DM_01236] [SWS_DM_01238] [SWS_DM_01240] [SWS_DM_01241] [SWS_DM_01243] [SWS_DM_01244] [SWS_DM_01245] [SWS_DM_01246] [SWS_DM_01247] [SWS_DM_01248] [SWS_DM_01249] [SWS_DM_01251] [SWS_DM_01360] [SWS_DM_01961] [SWS_DM_01962] [SWS_DM_01967] [SWS_DM_01968] [SWS_DM_01969] [SWS_DM_01970] [SWS_DM_02077]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04256]	Support of SOVD	[SWS_DM_01369] [SWS_DM_01370] [SWS_DM_01371] [SWS_DM_01372] [SWS_DM_01373] [SWS_DM_01374] [SWS_DM_01450] [SWS_DM_01451] [SWS_DM_01452] [SWS_DM_01453] [SWS_DM_01454] [SWS_DM_01455] [SWS_DM_01457] [SWS_DM_01458] [SWS_DM_01459] [SWS_DM_01460] [SWS_DM_01461] [SWS_DM_01462] [SWS_DM_01463] [SWS_DM_01464] [SWS_DM_01465] [SWS_DM_01466] [SWS_DM_01467] [SWS_DM_01468] [SWS_DM_01821] [SWS_DM_01822] [SWS_DM_01823] [SWS_DM_01824] [SWS_DM_01825] [SWS_DM_01826] [SWS_DM_01827] [SWS_DM_01828] [SWS_DM_01829]
[RS_Diag_04257]	Capability Description	[SWS_DM_01456] [SWS_DM_01786]
[RS_Diag_04258]	Discovering of Entities and Resources	[SWS_DM_01389] [SWS_DM_01404] [SWS_DM_01417] [SWS_DM_01420] [SWS_DM_01429] [SWS_DM_01442] [SWS_DM_01446] [SWS_DM_01449] [SWS_DM_01790] [SWS_DM_01797]
[RS_Diag_04259]	Fault Memory Access	[SWS_DM_01406] [SWS_DM_01409] [SWS_DM_01411] [SWS_DM_01412] [SWS_DM_01413] [SWS_DM_01414] [SWS_DM_01415] [SWS_DM_01416] [SWS_DM_01417] [SWS_DM_01418] [SWS_DM_01561] [SWS_DM_01567]
[RS_Diag_04260]	Read / Write Access	[SWS_DM_01419] [SWS_DM_01420] [SWS_DM_01430] [SWS_DM_01431] [SWS_DM_01432] [SWS_DM_01433] [SWS_DM_01434] [SWS_DM_01435] [SWS_DM_01436] [SWS_DM_01437] [SWS_DM_01439] [SWS_DM_01440] [SWS_DM_01441] [SWS_DM_01442] [SWS_DM_01444] [SWS_DM_01445] [SWS_DM_01446] [SWS_DM_01447] [SWS_DM_01448] [SWS_DM_01449]
[RS_Diag_04261]	Configuration Access	[SWS_DM_01421] [SWS_DM_01422] [SWS_DM_01423] [SWS_DM_01424] [SWS_DM_01425] [SWS_DM_01426] [SWS_DM_01427] [SWS_DM_01428] [SWS_DM_01429] [SWS_DM_01787] [SWS_DM_01788] [SWS_DM_01860] [SWS_DM_01861] [SWS_DM_01862] [SWS_DM_01863] [SWS_DM_01864] [SWS_DM_01865] [SWS_DM_01866] [SWS_DM_01867] [SWS_DM_01868] [SWS_DM_01869] [SWS_DM_01870] [SWS_DM_01871] [SWS_DM_01872] [SWS_DM_01930]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04262]	Control of Operations	[SWS_DM_01390] [SWS_DM_01391] [SWS_DM_01392] [SWS_DM_01393] [SWS_DM_01394] [SWS_DM_01395] [SWS_DM_01396] [SWS_DM_01397] [SWS_DM_01398] [SWS_DM_01399] [SWS_DM_01400] [SWS_DM_01401] [SWS_DM_01402] [SWS_DM_01403] [SWS_DM_01404] [SWS_DM_01405] [SWS_DM_01478] [SWS_DM_01479] [SWS_DM_01480] [SWS_DM_01481] [SWS_DM_01482] [SWS_DM_01491] [SWS_DM_01492] [SWS_DM_01493] [SWS_DM_01494] [SWS_DM_01495]
[RS_Diag_04263]	Support of Target Modes	[SWS_DM_01379] [SWS_DM_01380] [SWS_DM_01381] [SWS_DM_01382] [SWS_DM_01383] [SWS_DM_01384] [SWS_DM_01385] [SWS_DM_01386] [SWS_DM_01387] [SWS_DM_01388] [SWS_DM_01389]
[RS_Diag_04264]	SOVD specific Locking/Semaphore mechanism	[SWS_DM_01375] [SWS_DM_01443] [SWS_DM_01473] [SWS_DM_01474] [SWS_DM_01475] [SWS_DM_01476] [SWS_DM_01477] [SWS_DM_01929]
[RS_Diag_04265]	Software Update	[SWS_DM_01796] [SWS_DM_01797] [SWS_DM_01798] [SWS_DM_01799] [SWS_DM_01800] [SWS_DM_01801] [SWS_DM_01802] [SWS_DM_01803] [SWS_DM_01804] [SWS_DM_01874] [SWS_DM_01875] [SWS_DM_01876] [SWS_DM_01877] [SWS_DM_01878] [SWS_DM_01879] [SWS_DM_01881] [SWS_DM_01882] [SWS_DM_01883] [SWS_DM_01884] [SWS_DM_01885] [SWS_DM_01886] [SWS_DM_01887] [SWS_DM_01888] [SWS_DM_01889] [SWS_DM_01890] [SWS_DM_01891] [SWS_DM_01893] [SWS_DM_01894] [SWS_DM_01895] [SWS_DM_01896] [SWS_DM_01898] [SWS_DM_01899] [SWS_DM_01900] [SWS_DM_01901] [SWS_DM_01902] [SWS_DM_01903] [SWS_DM_01904] [SWS_DM_01905] [SWS_DM_01906] [SWS_DM_01907] [SWS_DM_01908] [SWS_DM_01909] [SWS_DM_01910] [SWS_DM_01911] [SWS_DM_01912] [SWS_DM_01913] [SWS_DM_01914] [SWS_DM_01915] [SWS_DM_01916] [SWS_DM_01917] [SWS_DM_01918] [SWS_DM_01919] [SWS_DM_01920] [SWS_DM_01921] [SWS_DM_01922] [SWS_DM_01923] [SWS_DM_01924] [SWS_DM_01925]

▽

△

Requirement	Description	Satisfied by
[RS_Diag_04266]	Bulk data	[SWS_DM_01789] [SWS_DM_01790] [SWS_DM_01791] [SWS_DM_01792] [SWS_DM_01793] [SWS_DM_01794] [SWS_DM_01795] [SWS_DM_01828] [SWS_DM_01829] [SWS_DM_01830] [SWS_DM_01831] [SWS_DM_01832] [SWS_DM_01833] [SWS_DM_01834] [SWS_DM_01835] [SWS_DM_01836] [SWS_DM_01837] [SWS_DM_01838] [SWS_DM_01839] [SWS_DM_01840] [SWS_DM_01841] [SWS_DM_01842] [SWS_DM_01843] [SWS_DM_01844] [SWS_DM_01845] [SWS_DM_01846] [SWS_DM_01847] [SWS_DM_01848] [SWS_DM_01849] [SWS_DM_01850] [SWS_DM_01851] [SWS_DM_01852] [SWS_DM_01853] [SWS_DM_01854] [SWS_DM_01855] [SWS_DM_01856] [SWS_DM_01857] [SWS_DM_01858] [SWS_DM_01873] [SWS_DM_01926]
[RS_Diag_04267]	Logging	[SWS_DM_01928]
[RS_Diag_04268]	Authentication	[SWS_DM_01376] [SWS_DM_01469] [SWS_DM_01470] [SWS_DM_01471] [SWS_DM_01472] [SWS_DM_01483] [SWS_DM_01484] [SWS_DM_01485] [SWS_DM_01486] [SWS_DM_01487] [SWS_DM_01488] [SWS_DM_01489] [SWS_DM_01490] [SWS_DM_01781] [SWS_DM_01782] [SWS_DM_01783] [SWS_DM_01819] [SWS_DM_01820] [SWS_DM_01927]
[RS_Diag_04270]	Notify applications about the changes of enable conditions	[SWS_DM_01094] [SWS_DM_01740]
[RS_Diag_04273]	SOVD service request validation	[SWS_DM_01466] [SWS_DM_01467] [SWS_DM_01468] [SWS_DM_01784] [SWS_DM_01785] [SWS_DM_01821] [SWS_DM_01827]
[RS_Ids_00810]	Basic SW security events	[SWS_DM_02014] [SWS_DM_02015] [SWS_DM_02016] [SWS_DM_02017] [SWS_DM_02018] [SWS_DM_02019] [SWS_DM_02020] [SWS_DM_02021] [SWS_DM_02022] [SWS_DM_02023] [SWS_DM_02024] [SWS_DM_02025] [SWS_DM_02026] [SWS_DM_02027] [SWS_DM_02028] [SWS_DM_02029] [SWS_DM_02030] [SWS_DM_02031] [SWS_DM_02032] [SWS_DM_02033] [SWS_DM_02034] [SWS_DM_02035] [SWS_DM_02036] [SWS_DM_02037] [SWS_DM_02038] [SWS_DM_02039] [SWS_DM_02040] [SWS_DM_02041] [SWS_DM_02042] [SWS_DM_02043] [SWS_DM_02044] [SWS_DM_02045] [SWS_DM_02046] [SWS_DM_02047] [SWS_DM_02048] [SWS_DM_02049] [SWS_DM_02050] [SWS_DM_02051] [SWS_DM_02052] [SWS_DM_02053] [SWS_DM_02054] [SWS_DM_02055] [SWS_DM_02056] [SWS_DM_02057] [SWS_DM_02058]

▽

△

Requirement	Description	Satisfied by
[RS_Main_00011]	Mechanisms for Reliable Systems	[SWS_DM_01571] [SWS_DM_01572] [SWS_DM_01573] [SWS_DM_01574]
[RS_Main_01002]	AUTOSAR shall support service-oriented communication	[SWS_DM_00557] [SWS_DM_00595] [SWS_DM_00600] [SWS_DM_00601] [SWS_DM_00603] [SWS_DM_00604] [SWS_DM_00617]

Table 6.1: Requirements Tracing

7 Functional specification

The functionality of DM is split into two layers: the UDS Transport Layer and the Application Layer. On the UDS Transport Layer, DM handles connections to Diagnostic Clients via standardized or user defined UDS Transport Protocols, see section 7.1 for details. The subcomponent of DM implementing a particular Transport Protocol is called a Transport Protocol Handler.

Besides UDS Transport Layer also communication via SOVD is supported according to [4]. The SOVD Transport Layer is covered in section 7.2 "SOVD Transport Layer".

On the Application Layer, DM implements the two main building blocks of diagnostics: Diagnostic Event Management and Diagnostic Communication Management, both according to UDS ISO 14229-1[1] and SOVD[4]. On AUTOSAR adaptive platform the Application Layer can be split into multiple SoftwareClusters, each with its own diagnostic address. Accordingly, DM instantiates for each SoftwareCluster a Diagnostic Server that implements diagnostics with scope given by this SoftwareCluster, see section 7.3. In context of SOVD the diagnostic address corresponds to an SOVD subcomponent, see section 7.3 "Diagnostic Server".

The link between the UDS Transport Layer and the Application Layer is implemented by the Transport Protocol Manager (see Section 7.1 "UDS Transport Layer".), which dispatches UDS messages in both directions: UDS requests from Diagnostic Clients are forwarded to the respective responsible Diagnostic Server instance, and UDS responses created by Diagnostic Server instances are dispatched towards the respective Transport Protocol Handler (see Section 7.1 " UDS Transport Layer".) that handles the connection to the Diagnostic Client. Accordingly, the dispatching between SOVD Transport Layer and different SOVD subcomponents need to be handled.

A broad subcomponent view on DM is given as follows:

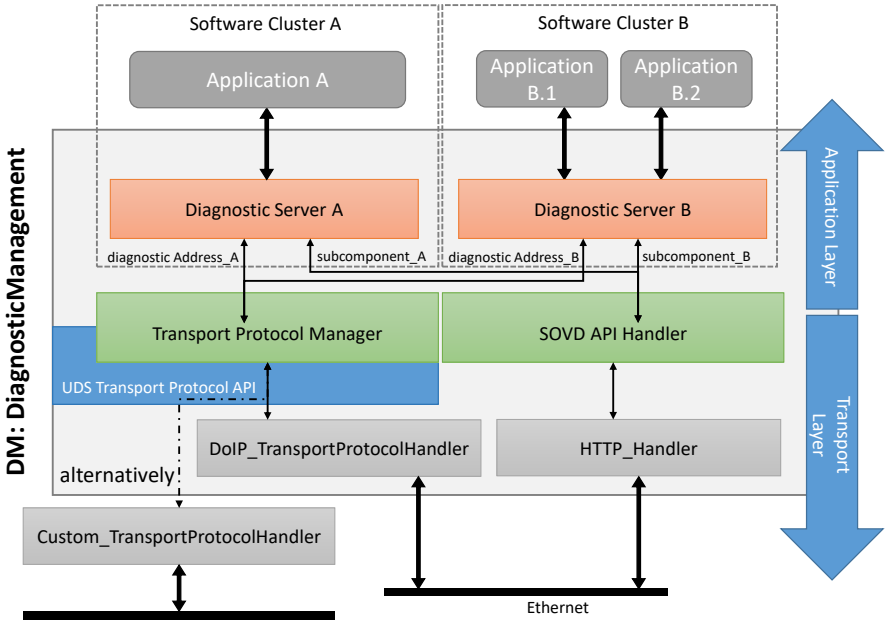

Figure 7.1: Component view on Diagnostic Management

7.1 UDS Transport Layer

Since there exist standardized as well as OEM specific UDS Transport Layers, the DM supports a standardized C++ API (called `Transport Protocol API`), where different kinds of UDS Transport Layers can be connected. Currently the Adaptive Platform only provides a detailed description of Ethernet-based network technologies, which mandates support of DoIP [2]. It is very likely, that upcoming releases of the DM will also detail CAN, CAN-FD, FR, ... networks. The `Transport Protocol API` allows for extensions of DM towards not-yet-detailed and proprietary UDS Transport Protocols.

The UDS Transport Protocol (TP) implementation can optionally provide the so far received UDS message content in `payloadInfo`. It is up to the TP implementation to decide how much of the so far received data is provided. It is recommended to provide at least the first two bytes of the received message. The DM is then able to detect a functional received tester present (UDS bypass logic) and accept this message, even the DM is currently processing a physical request. If the TP does not provide any data in `payloadInfo`, the DM will not be able to detect the functional TP and will reject the message reception in `apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` (see also [SWS_DM_00386]). This might or might not result to failures on the TP. E.g. DoIP provides 'Negative Acknowledge' (NACK) codes in that case, while CAN (Controller Area Network) does not have any TP failures.

7.1.1 Support of UDS Transport Layer

The UDS Transport Protocol API is formally described in section C. This section describes the required interaction of the components using this API. Each (proprietary) UDS Transport Protocol implementation subclasses the abstract class apext::diag::uds_transport::UdsTransportProtocolHandler, which shall be provided by DM according to [SWS_DM_00315].

7.1.1.1 Initialization, Starting and Stopping of a UDS TransportLayer

[SWS_DM_00329] Lifecycle management of an UDS Transport Protocol implementation

Upstream requirements: RS_Diag_04168

⌈The lifecycle of an UDS Transport Protocol implementation shall be managed by the DM in the following order:

- Creation of the UDS Transport Protocol implementation by calling its constructor apext::diag::uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler.
- Initializing of the UDS Transport Protocol implementation by calling apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize.
- Starting of the UDS Transport Protocol implementation by calling apext::diag::uds_transport::UdsTransportProtocolHandler::Start.
- Stopping of the UDS Transport Protocol implementation by calling apext::diag::uds_transport::UdsTransportProtocolHandler::Stop.

⌋

[SWS_DM_00330] Construction of an UDS Transport Protocol implementation

Upstream requirements: RS_Diag_04168

⌈The DM shall call the specific constructor of the UDS Transport Protocol implementation, where the argument handlerId is unique among all by DM instantiated UDS Transport Protocol implementations and the argument transportProtocolMgr is set to the reference of the instance of apext::diag::uds_transport::UdsTransportProtocolMgr provided by DM.⌋

[SWS_DM_00331] Initialization of an UDS Transport Protocol implementation

Upstream requirements: RS_Diag_04168

⌈The DM shall call the apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize during startup/initialization phase, before reporting ApplicationState.kRunning to the execution management.⌋

[SWS_DM_01746] Required successful initialization

Upstream requirements: RS_Diag_04168

⌈The DM shall call any method except of GetPeriodicHandler on apext::diag::uds_transport::UdsTransportProtocolHandler only after apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize has returned kInitializeOk.⌋

[SWS_DM_01745] Behavior on failed transport protocol initialization

Upstream requirements: RS_Diag_04168

⌈If a apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize returns kInitializeFailed, the DM shall stop the initialization process of the transport protocol and call no further methods on that apext::diag::uds_transport::UdsTransportProtocolHandler.⌋

[SWS_DM_01744] Processing of ChannelReestablished

Upstream requirements: RS_Diag_04168

⌈The DM shall process a call of apext::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished only after the call of apext::diag::uds_transport::UdsTransportProtocolHandler::Start.⌋

[SWS_DM_01743] Require a started UdsTransportProtocolHandler

Upstream requirements: RS_Diag_04168

⌈The DM shall call apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit and apext::diag::uds_transport::UdsTransportProtocolHandler::Stop only after apext::diag::uds_transport::UdsTransportProtocolHandler::Start was called.⌋

[SWS_DM_00332] Starting of an UDS Transport Protocol implementation

Upstream requirements: RS_Diag_04168

⌈The DM shall call the apext::diag::uds_transport::UdsTransportProtocolHandler::Start method of the UDS Transport Protocol implementation during startup/initialization phase, before reporting ApplicationState.kRunning to the execution management and after call to apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize has returned.⌋

[SWS_DM_01742] Asynchronous UDS protocol start

Upstream requirements: RS_Diag_04168

⌈The apext::diag::uds_transport::UdsTransportProtocolHandler::Start implementation shall be asynchronous and can continue starting the UDS protocol handler after the call has returned.⌋

[SWS_DM_00333] Stopping of an UDS Transport Protocol implementation

Upstream requirements: RS_Diag_04168

⌈The DM shall call the apext::diag::uds_transport::UdsTransportProtocolHandler::Stop method of each UDS Transport Protocol implementation it has started, if it is switching to state ApplicationState.kTerminating.⌋

[SWS_DM_01741] No more method calls after stop

Upstream requirements: RS_Diag_04168

⌈After apext::diag::uds_transport::UdsTransportProtocolHandler::Stop has returned, the handler-plugin shall NOT call to apext::diag::uds_transport::UdsTransportProtocolMgr with any other method except of apext::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped.⌋

[SWS_DM_00340] Waiting for Stop confirmation

Upstream requirements: RS_Diag_04168

⌈After having called apext::diag::uds_transport::UdsTransportProtocolHandler::Stop method of any UDS Transport Protocol implementation, it shall wait for the corresponding apext::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped callback with the related handlerId, before it finally terminates the process.⌋

7.1.1.2 UDS message reception on a UDS TransportLayer

[SWS_DM_00347] Channel identification in Indication

Upstream requirements: RS_Diag_04168

⌈UDS Transport Protocol implementation shall determine a distinct identifier to identify the network specific channel over which the UDS request has been received, which can be later used to deliver the UDS response to the source of the UDS request.⌋

Note: A diagnostic client has basically two address parts which together serve for its unique identification:

- The UDS source address (SA) in the clients/testers request which represent a technology/transport layer independent part.
- The technology/transport layer specific/dependent network endpoint source address, from which the request from the client originates. In Ethernet-based networks this typically is an IP-address/port number pair, while in CAN networks it is the CAN identifier of the CAN-TP message used by the client. In UDS on CAN (ISO ISO-15765-2[10]) contrary to DoIP, the SA is not explicitly transmitted, but directly deduced from the CAN identifier of the CAN-TP message. That means on CAN we do not have two separate address parts, only the network endpoint source address part is used for identification.

The side effect of this is that from the viewpoint of Diagnostic Server, which supports parallel Diagnostic Clients, it is a perfectly valid scenario that two Diagnostic Clients with the same UDS SA can be active in parallel if they originate from different/distinguishable network endpoints.

[SWS_DM_00385] Acceptance of UDS message reception

Upstream requirements: RS_Diag_04168

⌈If the DM is able to process the indicated request, apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage shall return a std::pair with apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult set to kIndicationOk and a apext::diag::uds_transport::UdsMessagePtr, which owns a valid apext::diag::uds_transport::UdsMessage object, with a capacity of so many bytes, the DM wants to process of the indicated request. The minimum size of the apext::diag::uds_transport::UdsMessage object shall be one byte.⌋

Note: For details about std::pair see [SWS_DM_00309].

[SWS_DM_00392] Properties of returned UdsMessage

Upstream requirements: RS_Diag_04168

⌈If the DM accepted the UDS message reception, the returned apext::diag::uds_transport::UdsMessage owned by apext::diag::uds_transport::UdsMessagePtr shall return a apext::diag::uds_transport::ByteSpan from apext::diag::uds_transport::UdsMessage::GetPayload, which shall be empty (i.e. empty() returns true, size() returns 0).⌋

Note: In the normal case, where DM accepts the complete UDS request for processing, it will provide a std::pair with apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult set to kIndicationOk and a apext::diag::uds_transport::UdsMessagePtr, which owns a valid apext::diag::uds_transport::UdsMessage object, with the capacity equal (or greater) to parameter size indicated by the UDS Transport Protocol implementation. There are use

cases (typically for negative responses), where the DM does NOT need the entire UDS request message data to generate the UDS response and therefore might return a apext::diag::uds_transport::UdsMessagePtr, which owns a valid apext::diag::uds_transport::UdsMessage object, with a capacity smaller than the indicated parameter size. E.g. this is useful e.g. in the case, where DM is busy and wants to ignore/reject a second parallel request. For declining a second request WITH sending a negative response according to [SWS_DM_00049], the DM would return an apext::diag::uds_transport::UdsMessagePtr with only enough capacity to be able to construct a valid negative response.

[SWS_DM_00386] Ignoring UDS message reception because DM is busy

Upstream requirements: RS_Diag_04168

⌈If the Transport Protocol Manager is calling apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage and the DM is busy due to the active processing of a service from the same Tester Present conversation and the Tester Present request is not a functional request with the optional provided payloadInfo '0x3E 0x80', the DM shall return a std::pair with apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult set to kIndicationOccupied and a apext::diag::uds_transport::UdsMessagePtr equal to nullptr.⌋

Functional TesterPresents (compare [SWS_DM_00126]) with suppressPosRspMsgIndicationBit = TRUE are exceptional requests in UDS and valid at any point in time. Therefore the DM allows to check for functional received TesterPresent request in the apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage. For handling of the functional TesterPresent, see chapter 7.3.2.8.23.

Note: For details about std::pair see [SWS_DM_00309].

Note: For declining/ignoring a second request without sending a negative response according to [SWS_DM_00290], the DM would choose this behavior.

[SWS_DM_00387] Ignoring UDS message reception because DM has no (memory) ressources

Upstream requirements: RS_Diag_04168

⌈If the DM is not able to process the indicated UDS request, because it has not enough (memory) resources to hold the indicated UDS request, it shall return a std::pair with apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult set to kIndicationOverflow and a apext::diag::uds_transport::UdsMessagePtr equal to UdsMessagePtr(nullptr).⌋

Note: For details about std::pair see [SWS_DM_00309].

Note: There might exist UDS Transport Protocol implementations, which make NO distinction between [SWS_DM_00386] and [SWS_DM_00387]. I.e. regardless, whether the DM returns a kIndicationOverflow or kIndicationOccupied, the behavior on transport layer level is the same. But, for instance, a CanTP UDS Transport Protocol implementation, would explicitly react on a kIndicationOverflow with sending a FC.OFLW on CanTP level to the UDS request sender.

[SWS_DM_00487] Ignoring UDS message reception because of unknown target address

Upstream requirements: RS_Diag_04168

⌈If the DM is not able to process the indicated UDS request, because the indicated target address is unknown to DM, it shall return a std::pair with apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult set to kIndicationUnknownTargetAddress and a apext::diag::uds_transport::UdsMessagePtr equal to UdsMessagePtr(nullptr).⌋

Note: For details about std::pair see [SWS_DM_00309].

[SWS_DM_00388] Filling provided UdsMessage

Upstream requirements: RS_Diag_04168

⌈If the DM returned kIndicationOk from the apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage, the UDS Transport Protocol implementation shall fill the apext::diag::uds_transport::UdsMessage owned by apext::diag::uds_transport::UdsMessagePtr from the received UDS request starting from SID up to either apext::diag::uds_transport::UdsMessage full capacity or up to the entire received UDS request message, whatever happens first.⌋

[SWS_DM_00345] Forwarding of UDS message

Upstream requirements: RS_Diag_04168

⌈If the UDS Transport Protocol implementation has filled the payload of the returned apext::diag::uds_transport::UdsMessagePtr, it shall call apext::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage on its apext::diag::uds_transport::UdsTransportProtocolMgr reference ((see [SWS_DM_00330]) with the returned apext::diag::uds_transport::UdsMessagePtr as argument.⌋

[SWS_DM_00389] Skipping Forwarding of UDS message

Upstream requirements: RS_Diag_04168

⌈If the DM returned a apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult NOT equal to kIndicationOk from the apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage,

the UDS Transport Protocol implementation shall NOT call apext::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage.⌋

[SWS_DM_00346] Aborting of UDS message

Upstream requirements: RS_Diag_04168

⌈If the UDS Transport Protocol implementation has already called apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage (see [SWS_DM_00342]), but is not willing to call apext::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage (maybe due to errors receiving the entire/remaining UDS request), it shall notify DM by calling apext::diag::uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure on its apext::diag::uds_transport::UdsTransportProtocolMgr reference ((see [SWS_DM_00330]) with the returned apext::diag::uds_transport::UdsMessagePtr as argument.⌋

[SWS_DM_00342] Indication of UDS message reception

Upstream requirements: RS_Diag_04168

⌈UDS Transport Protocol implementation shall call apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage on its apext::diag::uds_transport::UdsTransportProtocolMgr reference ((see [SWS_DM_00330])), as soon as it has at least the following information of an incoming UDS request available:

- UDS source address of the request.
- UDS target address of the request.
- Type of the UDS target address (physical or functional)
- Size of the entire UDS message starting from SID
- If the UDS payload is larger than 1 byte, at least two bytes are received and shall be forwarded in the parameter payloadInfo

⌋

7.1.1.3 UDS message transmission on a UDS TransportLayer

[SWS_DM_00348] Transmission of UDS response message

Upstream requirements: RS_Diag_04168

⌈DM shall send a diagnostic response UDS message to the same UDS Transport Protocol implementation, where it has received the UDS request message (see

[SWS_DM_00345]) by calling the apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit method of the UDS Transport Protocol implementation.⌋

[SWS_DM_00349] Reuse channel identifier of Indication

Upstream requirements: RS_Diag_04168

⌈DM shall set the argument channelId in the apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit call to the same value as in the Indication of the corresponding UDS request message (see [SWS_DM_00347]).⌋

[SWS_DM_00350] Confirmation of UDS message transmission

Upstream requirements: RS_Diag_04168

⌈When the UDS Transport Protocol implementation has a final feedback of the network layer, whether the UDS message triggered for transmission (see [SWS_DM_00348]) could be sent on the network or not, it shall notify DM by calling apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation ((see [SWS_DM_00330]) setting the message argument to the message parameter of the apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit call ([SWS_DM_00348]).⌋

[SWS_DM_00351] Confirmation Result

Upstream requirements: RS_Diag_04168

⌈When the the network layer was able to send the UDS response message to the network, the result argument in the apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation shall be set to kTransmitOk, otherwise to kTransmitFailed.⌋

7.1.1.4 Channel Notifications

Each incoming UDS request message is assigned an exact UDS Transport Protocol implementation specific Channel. With the normal request/reply paradigm in diagnostics, the UDS response message is sent out at the same Channel, from which the UDS request has been received. Therefore the Channel identifier is given to the DM in apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage in the form of parameter globalChannelId. The Channel part from this parameter is then used in the corresponding response in apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit.

There are use cases, where a diagnostic request might be answered deferred after the restart of the DM. The UDS service for ECU reset is a candidate for such a requirement. The upcoming requirements shall cover this use case.

[SWS_DM_00356] Requesting Notification of a channel reestablishment

Upstream requirements: RS_Diag_04168

⌈The DM shall call the apext::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment method of a UDS Transport Protocol implementation, with the parameter channelId set to the identifier of the Channel, where it needs a re-establishment notification.⌋

[SWS_DM_00357] Validity/lifetime of a Notification Request

Upstream requirements: RS_Diag_04168

⌈A notification request registered at a UDS Transport Protocol implementation according to [SWS_DM_00356] is valid only for the next call to apext::diag::uds_transport::UdsTransportProtocolHandler::Start until the following call to apext::diag::uds_transport::UdsTransportProtocolHandler::Stop of this UDS Transport Protocol implementation.⌋

[SWS_DM_00358] Notification of a channel reestablishment

Upstream requirements: RS_Diag_04168

⌈UDS Transport Protocol implementation shall call apext::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished on its UdsTransportProtocolMgr reference ((see [SWS_DM_00330]) setting the globalChannelId argument to the tuple consisting of its own handlerIdent and the ChannelID it has received in NotifyReestablishment (see [SWS_DM_00356]) once, in case it detects, that the underlying network Channel represented by ChannelID is getting available again.⌋

[SWS_DM_00359] Persistent Storage of Notification Request

Upstream requirements: RS_Diag_04168

⌈UDS Transport Protocol implementation shall store the notification request (see [SWS_DM_00356]) persistently, to be able to fulfill the notification even after a DM restart.⌋

7.1.2 DoIP

[SWS_DM_00475] Support of DoIP based on ISO 13400-2

Upstream requirements: RS_Diag_04242, RS_Diag_00025, RS_Diag_00027, RS_Diag_00028, RS_Diag_00081, RS_Diag_00083, RS_Diag_00084

⌈The DM shall support DoIP ISO 13400-2[2] specification.⌋

Note: According to the ISO 13400-2[2] specification, the DoIP entity supports protocol version = 0xFF in the vehicle identification request message.

[SWS_DM_00449] Supported DoIP message types

Upstream requirements: RS_Diag_00026, RS_Diag_00080, RS_Diag_00082, RS_Diag_00083, RS_Diag_00084, RS_Diag_00027

⌈The DM shall support the DoIP message types listed in Table [SWS_DM_01568].⌋

[SWS_DM_01568] supportedDoIpMessageTypes

Upstream requirements: RS_Diag_00026, RS_Diag_00080, RS_Diag_00082, RS_Diag_00083, RS_Diag_00084, RS_Diag_00027

⌈

Payload type value	Payload type Name
0x0000	Generic DoIP header negative acknowledge
0x0001	Vehicle identification
0x0002	Vehicle identification request message with EID
0x0003	Vehicle identification request message with VIN
0x0004	Vehicle announcement message/vehicle identification response message
0x0005	Routing activation request
0x0006	Routing activation response
0x0007	Alive check request
0x0008	Alive check response
0x4001	DoIP entity status request
0x4002	DoIP entity status response
0x4003	Diagnostic power mode information request
0x4004	Diagnostic power mode information response
0x8001	Diagnostic message
0x8002	Diagnostic message positive acknowledgement
0x8003	Diagnostic message negative acknowledgement

⌋

[SWS_DM_00855] Providing the VIN in DoIP protocol messages

Upstream requirements: RS_Diag_00026

⌈DM shall retrieve the VIN required in some DoIP messages by reading the data from the DID with DiagnosticDataIdentifier.representsVin set to TRUE.⌋

[SWS_DM_00814] Providing the PowerMode in DoIP protocol messages

Upstream requirements: RS_Diag_00026, RS_Diag_00080

⌈If the DM needs to know the PowerMode to be able to react or answer on any DoIP message, it shall obtain it by calling the method ara::diag::DoIPPowerMode::GetDoIPPowerMode.⌋

[SWS_DM_01525] `ara::diag::DoIPPowerMode` not yet offered when client requests DoIP PowerMode

Upstream requirements: RS_Diag_00080

⌈If a client requests the DoIP power mode information, the DM shall send a DoIP power mode information response with power mode value 0 ("Not ready") as long as the Offer function of the `ara::diag::DoIPPowerMode` interface is not called yet by the Adaptive Application that provides it (as defined in the ISO 13400-2[2] specification).⌋

[SWS_DM_00813] Providing the `GID` in `DoIP` protocol messages using application interface

Upstream requirements: RS_Diag_00026

⌈If `DoIpInstantiation.gid` is not configured, then DM shall obtain the GID by calling the method `ara::diag::DoIPGroupIdentification::GetGidStatus`.⌋

[SWS_DM_02008] Providing the `GID` if application interface not available ⌈If DM fails to obtain the GID according to [SWS_DM_00813], then DM shall send the GID with invalidity pattern with value as "0xFF 0xFF 0xFF 0xFF 0xFF 0xFF".⌋

[SWS_DM_02009] Providing the `GID` using configured value ⌈If `DoIpInstantiation.gid` is configured, then DM shall send the GID with value as configured in `DoIpInstantiation.gid`.⌋

[SWS_DM_02010] Providing `VIN`/`GID` status byte ⌈If `DoIpNetworkConfiguration.vehicleIdentificationSyncStatus` is configured to true, the DM shall provide the optional VIN/GID sync status according to⌋

[SWS_DM_02011] `VIN`/`GID` status on successful `VIN` retrieval ⌈If DM successfully retrieves VIN according to [SWS_DM_00855], the value of the "VIN/GID status" byte shall be 0x00.⌋

[SWS_DM_02012] `VIN`/`GID` status on unsuccessful `VIN` and `GID` synchronization ⌈If DM fails to retrieve VIN according to [SWS_DM_00855] and GID is not synchronized according to [SWS_DM_00813], the value of "VIN/GID status" byte shall be 0x10.⌋

[SWS_DM_02013] `VIN`/`GID` status on unsuccessful `VIN` and successful `GID` synchronization ⌈If DM fails to retrieve VIN according to [SWS_DM_00855] and GID is synchronized according to [SWS_DM_00813] or [SWS_DM_02008] Providing the GID using configured value, the value of "VIN/GID status" byte shall be 0x00.⌋

[SWS_DM_01527] `ara::diag::DoIPGroupIdentification` not yet offered when DM needs to retrieve GID

Upstream requirements: RS_Diag_00026

⌈If the DM shall send the GID while the Offer function of the ara::diag::DoIP-GroupIdentification interface was not called yet by the Adaptive Application that provides it, the DM shall send a GID value equal to 0x00 0x00 0x00 0x00 0x00 0x00 or 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF ("value not set"), as defined in Table 5 and Table 1 in the ISO 13400-2[2] specification.⌋

[SWS_DM_00815] When to send Vehicle announcement messages on interfaces without activation line control

Upstream requirements: RS_Diag_04242

⌈The DM gets notified, when to send out vehicle announcement messages on a network interface without activation line control (isActivationLineDependent == FALSE) by a call to method ara::diag::DoIPTriggerVehicleAnnouncement::Trig-gerVehicleAnnouncement, which DM has to provide. The method call contains the network interface identified via networkInterfaceId on which the announcement shall be sent.⌋

[SWS_DM_00816] Notification of activation line status change on activation line controlled network interfaces

Upstream requirements: RS_Diag_04242

⌈The DM gets notified, when the activation line status changes for activation line controlled network interfaces (isActivationLineDependent == TRUE) via software components providing an instance of DiagnosticDoIPActivationLineIn-terface. The DM shall identify for which network interface an instance of Diag-nosticDoIPActivationLineInterface is providing the activation line status via call to method ara::diag::DoIPActivationLine::GetNetworkInterfaceId. Whenever the status of the activation line of the related network interface changes, the application calls ara::diag::DoIPActivationLine::UpdateActivation-LineState.⌋

The DM needs to instantiate a singleton of ara::diag::DoIPTriggerVehi-cleAnnouncement only. The ara::diag::DoIPTriggerVehicleAnnounce-ment::TriggerVehicleAnnouncement method gets the according network-InterfaceId passed, to which the DoIP Vehicle Announcement Message (see [SWS_DM_00449]) shall be sent. Different AAs monitoring the DoIP Activation Line will inform the DM about an Activation Line toggle on its monitoring networkInter-faceId.

[SWS_DM_01526] `ara::diag::DoIPActivationLine` not yet offered on activation line controlled network interfaces

Upstream requirements: RS_Diag_04242

⌈As long as the Offer function of the `ara::diag::DoIPActivationLine` interface is not called by some Adaptive Application for a given activation line controlled network interface (`DoIpNetworkConfiguration.isActivationLineDependent` == TRUE) the `DoIP` communication shall not be started on the related network interface, since the `DoIP` Activation Line state cannot be sensed by the `DM` (according to the ISO 13400-2[2] specification).⌋

[SWS_DM_02005] Providing `VIN` if interface is not available ⌈If `DM` fails to retrieve `VIN` according to [SWS_DM_00855], then `DM` shall return `VIN` with value as '0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF'⌋

[SWS_DM_01361] Providing `EID` in DoIP protocol messages using Application Interface

Upstream requirements: RS_Diag_00026

⌈If `DoIpNetworkConfiguration.eidRetrieval` is configured with `DoIpEidRetrievalEnum.eidUseApi`, `DM` shall obtain `EID` by calling the method `ara::diag::DoIPEntityIdentification::GetEntityId`.⌋

[SWS_DM_01528] `ara::diag::DoIPEntityIdentification` not yet offered when DM needs to retrieve `EID` from Application

Upstream requirements: RS_Diag_00026

⌈If `DM` fails to retrieve `EID` according to [SWS_DM_01361],the `DM` shall send 6-byte `EID` value equal '0xFF 0xFF 0xFF 0xFF 0xFF 0xFF'.⌋

[SWS_DM_02006] Providing `EID` using manually configured value ⌈If `DoIpNetworkConfiguration.eidRetrieval` is configured with `DoIpEidRetrievalEnum.eidUseConfigValue`, `DM` shall obtain the `EID` as configured in `DoIpInstantiation.eid`.⌋

[SWS_DM_02007] Providing `EID` using `MAC` of network interface ⌈If `DoIpNetworkConfiguration.eidRetrieval` is configured with `DoIpEidRetrievalEnum.eidUseMac`, `DM` shall obtain the `EID` by reading the `MAC` of network interface⌋

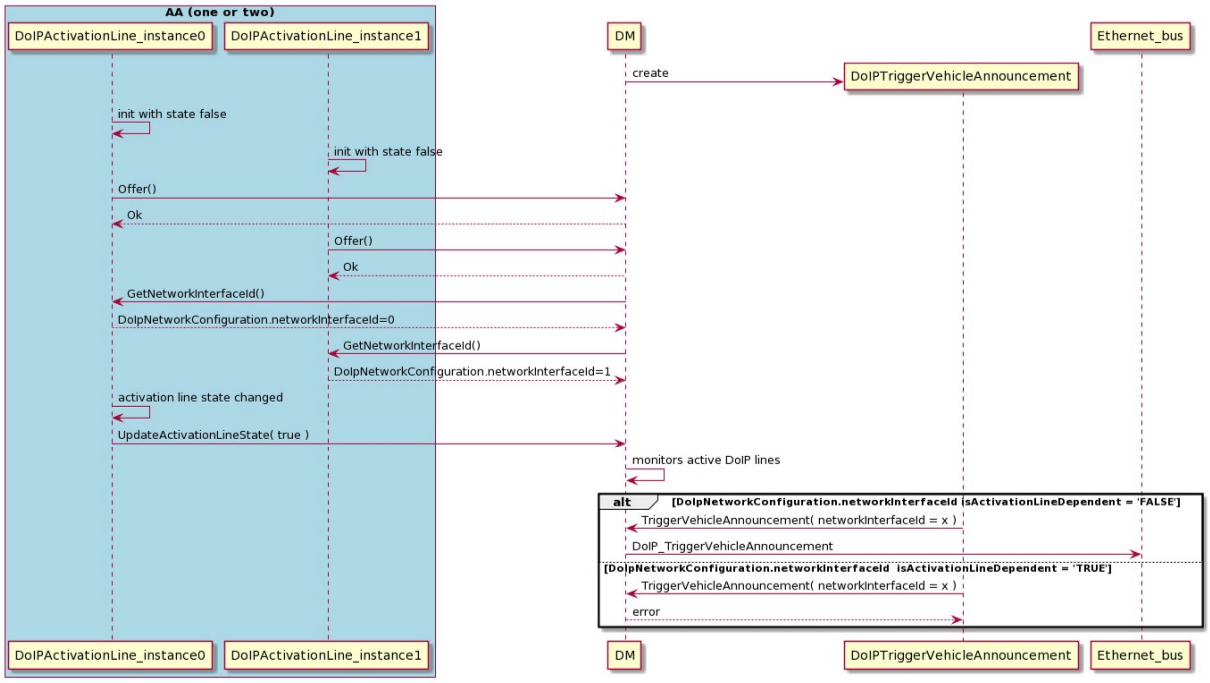

Figure 7.2: Example Sequence Diagram for Activation Line and Vehicle Announcement API use case

[SWS_DM_01979] Secure Communication for DoIP using TLS

Upstream requirements: RS_Diag_00140

⌈The Diagnostic Server instances shall secure its DoIP connections using TLS according ISO 13400-2[2] considering the configuration referenced by DoIpNetworkConfigurationDesign.networkConfiguration.secureComPropsForTcp.⌋

[SWS_DM_01980] Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.tcpInitialInactivityTime is not configured, the DiagnosticManager shall use a value of 2 seconds.⌋

[SWS_DM_01981] Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.tcpGeneralInactivityTime is not configured, the DiagnosticManager shall use a value of 300 seconds.⌋

[SWS_DM_01982] Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.vehicleAnnouncementCount is not configured, the DiagnosticManager shall use a value of 3.⌋

[SWS_DM_01983] Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.vehicleAnnouncementInterval is not configured, the DiagnosticManager shall use a value of 0.5 seconds.⌋

[SWS_DM_01984] Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.tcpAliveCheckResponseTimeout is not configured, the DiagnosticManager shall use a value of 0.5 seconds.⌋

[SWS_DM_01985] Default value for the attribute maxTesterConnections of meta-class DoIpNetworkConfiguration

Upstream requirements: RS_Diag_04166

⌈If the attribute DoIpNetworkConfiguration.maxTesterConnections is not configured, the DiagnosticManager shall use a value of '1'.⌋

[SWS_DM_01986] Used DoIP Protocol Version

Upstream requirements: RS_Diag_00025

⌈

The DoIP module shall derive its used DoIP protocol version from DoIpFunctionalClusterDesign.doIpProtocolVersion. If this value is not configured the DoIP protocol version 0x03 shall be used.⌋

The DoIP protocol version is used during checks for fitting protocol version during receiption of DoIP messages and all send messages are provided with this protocol version.

[SWS_DM_02002] Support of entity status response item Max. datasize (MDS)

Upstream requirements: RS_Diag_00082

⌈The "Max data size" bytes are only supported if the configuration parameter DoIpFunctionalClusterDesign.entityStatusMaxByteFieldUse is set to TRUE. In this case, the diagnostic entity status response message shall contain the configured DoIpFunctionalClusterDesign.maxRequestBytes in the "Max data size" field.⌋

7.1.3 Dispatching of UDS Requests

The Transport Protocol Manager has to dispatch the UDS-messages between the Transport Protocol Handler and the Diagnostic Server instances. To do this the Transport Protocol Manager uses the following information as provided by the Transport Protocol Handler indication function on received UDS requests:

- Target Address
- Target Address Type (phys / func)

In transmit direction the Transport Protocol Manager provides the UDS message from the Diagnostic Server and calls the Transmit method from the Transport Protocol Handler.

[SWS_DM_00390] Dispatching physical Request

Upstream requirements: RS_Diag_04216

⌈DM shall dispatch each UDS physical request to the Diagnostic Server instance responsible for the SoftwareCluster with diagnosticAddress matching the TargetAddress of the received UDS request and addressSemantics set to physicalAddress.⌋

[SWS_DM_00391] Dispatching functional Request

Upstream requirements: RS_Diag_04216

⌈DM shall dispatch each UDS functional request to all Diagnostic Server instances responsible for those SoftwareClusters with a diagnosticAddress matching the TargetAddress of the received UDS request and addressSemantics set to functionalAddress.⌋

7.2 SOVD Transport Layer

Beside the opportunity to perform diagnostic communication between a diagnostic client and the Diagnostic Manager via DoIP, the DM will provide further communication interfaces for SOVD-based communication. Therefore, the DM requires one or more socket connections including port and IP address to be able to establish a HTTP resp. HTTPS connection.

[SWS_DM_01369] DM as SOVD Server

Upstream requirements: RS_Diag_04256

⌈The DM shall act as SOVD Server as specified by ASAM SOVD[4] according to the configuration referenced by SovdModuleInstantiation.communicationConnector.⌋

[SWS_DM_01370] DNS-Based Service Discovery and Multicast DNS for SOVD

Upstream requirements: RS_Diag_04256

⌈The DM shall support DNS-SD (DNS-Based Service Discovery) and mDNS (Multicast DNS) as specified by ASAM SOVD[4].⌋

[SWS_DM_01371] Secure Communication for SOVD using TLS

Upstream requirements: RS_Diag_04256

⌈The DM shall secure its HTTP connections using TLS according to the configuration referenced by SovdModuleInstantiation.securePropsForTcp.⌋

[SWS_DM_01372] Representation of DM by SOVD component

Upstream requirements: RS_Diag_04256

⌈The DM shall be represented by an SOVD component that shall be a child of the SOVD Server in the role components. The corresponding component-identifier shall be derived from the SovdServerInstantiation.⌋

[SWS_DM_01373] Representation of Diagnostic Server instance by SOVD subcomponents

Upstream requirements: RS_Diag_04256

⌈Each Diagnostic Server instance shall be represented by one corresponding SOVD component that shall be a child of the SOVD component representing the DM in the role subcomponent. The corresponding component-identifier shall be derived from the SoftwareClusterSovdAddress.⌋

[SWS_DM_01374] Dispatching of SOVD requests/responses

Upstream requirements: RS_Diag_04256

⌈Within the DM SOVD requests and responses shall be dispatched to and from the Diagnostic Server instances based on the SOVD path.⌋

7.3 Diagnostic Server

The AUTOSAR adaptive platform is able to be extended with new software packages without re-flashing the entire ECU. The individual software packages are described by SoftwareClusters. To support the current approaches of diagnostic management (like software updates), two architecture approaches are possible:

- Each SoftwareCluster has its own diagnosticAddress and DiagnosticContributionSet.
- Multiple SoftwareCluster share the diagnosticAddress and also the DiagnosticContributionSet.

For details on the semantics and precise configuration of SoftwareClusters, see [11].

DM is intended to support an own Diagnostic Server instance per DiagnosticContributionSet. All Diagnostic Server instances share the same UDS TransportLayer (see Figure 7.1) and each Diagnostic Server manages its own resources.

[SWS_DM_00420] Instantiation of Diagnostic Server

Status: DRAFT
Upstream requirements: RS_Diag_04216

⌈DM shall instantiate an independent Diagnostic Server per DiagnosticContributionSet, which is referenced by one or multiple SoftwareClusters in the role of diagnosticExtract, with dedicated resources and functionality configured by this DiagnosticContributionSet.⌋

Details on required configuration items are described in section 7.3.5.

This chapter focuses on requirements concerning a single Diagnostic Server, hence we assume that

- requests from Diagnostic Clients are already dispatched towards this Diagnostic Server according to [SWS_DM_00390] and [SWS_DM_00391],
- DEXT configuration elements used in a requirement are meant to be part of the DiagnosticContributionSet associated to the Diagnostic Server according to [SWS_DM_00420].

In particular, we note that requests addressing different Diagnostic Server instances shall be processed independently by the respective Diagnostic Servers.

Note: An example of a method call with MetaInfo as parameter is [SWS_DM_00618]. In general the callee is not supposed to store or reuse the MetaInfo object after the DM function has returned. This would result in undefined behavior.

7.3.1 Interaction between DM and applications

This chapter introduces ara::diag and provides general information on how these interfaces behave and how they shall be used.

Most interaction between DM and application is realized by instances of ara::diag classes. The application instantiates the ara::diag objects and uses the provided interface to interact with the DM. The application itself can be deployed anywhere. They can run in the local SoftwareCluster, but also remote on other SoftwareClusters or even other hardware is possible.

[SWS_DM_01529] Behavior on failed ara::diag instantiation

Status: DRAFT
Upstream requirements: RS_AP_00158

⌈If a client application instantiates an ara::diag interface with an invalid instance specifier in the constructor, the DM shall treat it as a violation according to [SWS_CORE_90006] .⌋

Note: There are various reasons why an instantiation of an ara::diag interface can fail. Constructing the object requires valid PortPrototype, valid mapping and valid deployment configuration in the model. This needs to be checked in the constructor.

7.3.1.1 MetaInfo class

The ara::diag::MetaInfo class specifies a mechanism to provide meta informations, i.e. from transport protocol layer, to an interested application. To support this, ara::diag::MetaInfo::GetValue is specified, which provides the according value represented as a *ara::core::StringView*. The context of the current request may also be retrieved by an application by calling ara::diag::MetaInfo::GetContext. The context of a request could be either kDiagnosticCommunication, kFaultMemory or kDoIP. The detailed information on meanings of the context and key-value pairs can be looked up in Table 7.1 “MetaInfo type definition”.

[SWS_DM_01345] Lifetime of MetaInfo

Upstream requirements: RS_Diag_04170

⌈For each method called from DM to application with a MetaInfo as parameter, the DM shall limit the guaranteed life time of the MetaInfo object only to the time that function call is active.⌋

Note: An example of a method call with MetaInfo as parameter is [SWS_DM_00618]. In general the callee is not supposed to store or reuse the MetaInfo object after the DM function has returned. This would result in undefined behavior.

Key	Context		String-Format	Description
kSA	kDiagnostic-Communication		[0-9A-F]{4}	UDS Source Address from which the diagnostic request has been sent. The value is formatted as hexidecimal number. For example tester SA of decimal 240 will have the stringified value "00F0".
kTA	kDiagnostic-Communication		[0-9A-F]{4}	UDS Target Address to which the diagnostic request has been sent. The value is formatted as hexidecimal number. For example TA of decimal 59 will have the stringified value "003B".
kTAType	kDiagnostic-Communication		"PHYS" or "FUNC"	Indicator whether request is functional or physical addressed.
kRe-questHandle	kDiagnostic-Communica-tion, kSovd		[0-9]{1,5}	Key for the RequestHandle, which shall be identical for all API calls within the context of the same diagnostic request. E.g. for a Validate() and an final Confirmation() call in the context of the same diagnostic request, the same value for this key has to be placed in the metaInfo.
kLocalIP	kDiagnostic-Communication		IPv4 or IPv6 address	Key for the local IP address on which the current request gets received (this might be of interest in case the ECU is multi-homed and could receive diagnostic requests via DoIP on different interfaces). The value will be either a string in IPv4 address notation (decimal representation of address parts separated with ".") or a string in IPv6 notation (hexadecimal representation of address parts separated with ":" according to section 2.2 of RFC 4291
kLocalPort	kDiagnostic-Communication		[0-9]{1,5}	Key for the local port number on which the current request gets received. The value will be the stringified decimal representation of the port number.
kRemoteIP	kDiagnostic-Communication		IPv4 or IPv6 address	Key for the remote IP address from which the current request gets received. The value will be either a string in IPv4 address notation (decimal representation of address parts separated with ".") or a string in IPv6 notation (hexadecimal representation of address parts separated with ":" according to section 2.2 of RFC 4291)
kRemotePort	kDiagnostic-Communication		[0-9]{1,5}	Key for the remote port number from which the current request gets received. The value will be the stringified decimal representation of the port number.
kDtc		kFaultMemory	[0-9A-F]{6}	DTC number for which this interface is triggered
kBaseUri		kSOVD	URI	Base URI used for the request.
kEntityPath		kSOVD		Entity path used for the request.
kResourcePath		kSOVD		Path to the requested resource or resource collection.
kClientIdentity		kSOVD		Identity of the client. String format is defined by application handling the authentication.

Table 7.1: MetaInfo type definition

7.3.1.2 Concurrency of ara::diag interfaces

Some constructors of ara::diag interfaces accepts an parameter of type `ara::diag::ConcurrencyType` which allows an AA developer to inform the `DM` if the callback(s)

API is implemented in a thread-safe manner or not. If the ara::diag::ConcurrencyType is set to kConcurrent, then DM can invoke the API multiple times without waiting for the previous invocation to deliver the final result. In case the ara::diag::ConcurrencyType is set to kNotConcurrent, DM will block the request till the result of the ongoing request is fully delivered (Status of the future returned is ready).

The usage of ara::diag::ConcurrencyType is only limited to the APIs implemented by an AA. All the other APIs have their thread-safety specified in the API description.

7.3.1.3 Caching of application calls

An application can call a ara::diag method call at any point in time and independent from the current state of the DM process itself. For the application calling such a function it is unknown if the DM is actually currently available or not. Typical examples are:

- At startup the application is up and running and reporting information but the DM is not yet started and likely be up shortly after the call
- The communication between the ara::diag interface and the DM itself is temporarily interrupted

If the application would need to handle these situations to react on the not available DM, they would get very complex. Furthermore, each and every application would need to implement a complex strategy. To avoid this and to ease the interaction between application and DM for these functions, the ara::diag methods where the application calling the DM outside of service processing, the DM caches the relevant information and forwards this data to the DM instance once it is available. The application itself can simply concentrate on its function. Examples where this is applied are:

- Reporting states for enabled conditions
- Reporting state for clear DTC conditions
- Reporting state for monitor results

For all ara::diag methods that have kServiceNotAvailable as possible error code, the method will not cache the request and indicate the application that the method call has failed. This also means that all ara::diag methods that do not have kServiceNotAvailable as possible error codes, the ara::diag method is responsible to provide the requested data to the DM and the responsible action is delivered immediately to the DM or cached by the implementation of the ara::diag method.

7.3.2 Diagnostic Communication Management

A central element in the handling of diagnostic communication is the term Diagnostic Conversation, which is described in section 7.3.2.1. A UDS request is always

processed in the context of a Diagnostic Conversation. A single Diagnostic Server can handle multiple Diagnostic Conversations in parallel.

7.3.2.1 Diagnostic Conversations

A Diagnostic Conversation depicts a conversation between a distinct Diagnostic Client and a Diagnostic Server instance. In contrast to CP, on AP the details of connections between Diagnostic Clients and Diagnostic Server instances are not statically configured, but a Diagnostic Conversation is dynamically allocated during run-time of the Diagnostic Server instance.

For an incoming UDS request, the Diagnostic Server instance is identified via the target address of the UDS request (see [SWS_DM_00390], [SWS_DM_00391]), whereas the identification of the Diagnostic Client is transport layer specific.

[SWS_DM_00421] Identification of a Diagnostic Client

Upstream requirements: RS_Diag_04005

⌈The Diagnostic Server instance shall identify a Diagnostic Client by means of the tuple of sourceAddr and globalChannelId provided by the TP Layer on call of apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage, see [SWS_DM_00347].⌋

[SWS_DM_01259] Validation of Security Level Locked in ara::diag::Conversation::GetDiagnosticSecurityLevelShortName

Upstream requirements: RS_Diag_04005

⌈If the application calls the ara::diag::Conversation::GetDiagnosticSecurityLevelShortName with the parameter securityLevel set to kLocked, the Diagnostic Server instance shall return error code kInvalidArgument.⌋

[SWS_DM_01260] Validation of Invalid Security Level in ara::diag::Conversation::GetDiagnosticSecurityLevelShortName

Upstream requirements: RS_Diag_04005, RS_Diag_04227

⌈If the application calls ara::diag::Conversation::GetDiagnosticSecurityLevelShortName with the parameter securityLevel set to value that is not equal to any ((DiagnosticSecurityAccess.requestSeedId + 1) / 2) in the DiagnosticContributionSet, the Diagnostic Server instance shall return error code kInvalidArgument.⌋

[SWS_DM_01261] Validation of Invalid Session Level in `ara::diag::Conversation::GetDiagnosticSessionShortName`

Upstream requirements: RS_Diag_04005, RS_Diag_04226

⌈If the application calls `ara::diag::Conversation::GetDiagnosticSessionShortName` with the parameter `session` set to a value that is not equal to any configured `DiagnosticSession` in the `DiagnosticContributionSet`, the `Diagnostic Server instance` shall return error code `kInvalidArgument`.⌋

7.3.2.1.1 Multiple Client Handling

The DM is capable of parallel processing of client requests, as long as all clients are in the default session.

Note: The term "pseudo parallel concept" is already defined in ISO 14229-1, Annex J, and this possibility is explicitly limited to OBD in parallel with UDS protocol. The "pseudo parallel mode" allows other protocol combination can be processed in parallel. Particularly the use case of parallel processing of two or more UDS protocol requests.

Pseudo Parallel Mode In pseudo parallel mode, the DM is capable of parallel processing of client requests, as long as all clients are in the default session. If one client switches to a non-default session, the DM will process only diagnostic requests from the conversation of that tester which has requested the non-default session. The DM itself distinguishes between two kinds of parallel processing in default session: Fully parallel processing and sequential processing, where a concurrent access can be denied with an NRC '0x21 (busyRepeatRequest)' or an internal wait. In full parallel processing all application functions called from the DM would require a full reentrancy capability. Re-entrant software development is often by far more complex than non-reentrant software. The DM respects this situation and leaves it up to the application software developer to decide if a given application can be called re-entrant. A vehicle manufacturer may require that certain applications are supporting re-entrant functionality. The DM needs to evaluate the information of supported re-entrancy and behave accordingly. All required 'ara::diag' ports have a constructor parameter to tell the DM if a certain port allows re-entrant calls or not.

Impact of SOVD With the introduction of SOVD[4] not only parallel handling of multiple UDS(ISO 14229-1) clients needs to be considered but also SOVD must be taken into account. The `SOVD lock` mechanism needs to be treated specially here. The `SOVD lock` mechanism allows to gain exclusive access to a Diagnostic server. Thus, if a `SOVD lock` was acquired from any client, parallel access with multiple UDS (ISO 14229-1) or SOVD clients shall not be possible. For SOVD requests without an `SOVD lock` the same parallel execution rules as for ISO 14229-1 apply. Further details regarding parallel handling for SOVD are discussed in section 7.6.3.1.

[SWS_DM_00940] Concurrent ara::diag interface calls for service processing

Upstream requirements: RS_Diag_04166

⌈The DM shall only call a method on an interface for service processing in a concurrent way, if the DM is

- in default session and
- multiple conversation requests are being processed simultaneously and
- the requested diagnostic service requires that the DM is calling the same required port for different clients and
- the interface constructor parameter typed by ara::diag::ConcurrencyType is set to kConcurrentType

⌋

In any other case, the DM will not call a method re-entrant and behave according to ISO 14229-1. There are various ways how a DM can handle concurrent requests to the same resource. It is implementation specific, which solution is chosen. One of the options is to return an NRC '0x21 (busyRepeatRequest)', but also other means such as delaying the request might be used. The DM does explicitly provide the possibility to implement project specific solutions to meet the vehicle manufacturers diagnostic requirements.

[SWS_DM_00941] Concurrent ara::diag interface calls for DID read processing

Upstream requirements: RS_Diag_04166

⌈The DM shall only call a read DID processing method on an interface for service processing in a concurrent way if it is:

- in default session and
- multiple conversation requests are being processed simultaneously and
- the requested diagnostic service requires that the DM is calling methods on class namespacelistdataidentifier::dataidentifierinterfacename or ara::diag::GenericDataIdentifier and the same required port for different clients and
- the interface constructor has the parameter ara::diag::DataIdentifierConcurrencyType is set to kConcurrentType

⌋

[SWS_DM_00942] Concurrent ara::diag interface calls for DID write processing

Upstream requirements: RS_Diag_04166

⌈The DM shall only call a write DID processing method on an interface for service processing in a concurrent way if it is:

- in default session and
- multiple conversation requests are being processed simultaneously and
- the requested diagnostic service requires that the DM is calling methods on class namespacelistdataidentifier::dataidentifierinterfacename or ara::diag::GenericDataIdentifier and the same required port for different clients and
- the interface constructor has the parameter ara::diag::DataIdentifierConcurrencyType is set to kConcurrentType

⌋

[SWS_DM_00943] Concurrent ara::diag interface calls for DID read and write processing

Upstream requirements: RS_Diag_04166

⌈The DM shall only call a write and read DID processing method on an interface for service processing in a concurrent way if it is:

- in default session and
- multiple conversation requests are being processed simultaneously and
- the requested diagnostic service requires that the DM is calling methods on class namespacelistdataidentifier::dataidentifierinterfacename or ara::diag::GenericDataIdentifier and the same required port for different clients and
- the interface constructor has the parameter ara::diag::DataIdentifierConcurrencyType is set to kConcurrentType

⌋

[SWS_DM_00944] Validity of re-entrant ara::diag interface calls for DID processing

Upstream requirements: RS_Diag_04166

⌈If the DM is requested to perform a parallel call to a DID interface according to [SWS_DM_00941], [SWS_DM_00942] or [SWS_DM_00943] and the conditions for the parallel call are not fulfilled, the DM shall not call any method on that interface until any other call from the DM has returned.⌋

[SWS_DM_01375] Behavior on locked SOVD

Upstream requirements: RS_Diag_04264

⌈If a service cannot be performed or a session change is not possible due to SOVD lock, NRC 0x21 (BusyRepeatRequest) shall be returned.⌋

7.3.2.1.2 Life-cycle of a Diagnostic Conversation

The life-cycle of a Diagnostic Conversation starts with the first reception of a UDS request from the given Diagnostic Client to the Diagnostic Server instance and ends either if it is canceled (see section 7.3.2.9) or if **all** of the following conditions are satisfied:

- UDS request processing is finished by either
  - sending positive or final negative response and processing apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation call from TP-layer according to [SWS_DM_00350],
  - suppressing positive response,
  - suppressing negative response,
  - suppressing any response according to [SWS_DM_00860].
- associated Session is the Default Session.

Note: A Diagnostic Conversation in Non-Default Session is kept alive, as long as no Session time-out occurred. In this case, possibly multiple UDS requests are processed within this Lifecycle.

7.3.2.1.3 Diagnostic Conversation Service Interface

In some cases, the current state of a Diagnostic Conversation needs to be known by some Adaptive Applications. For this purpose, the Diagnostic Server instance provides instances of the Service Interface ara::diag::Conversation.

[SWS_DM_00840] Instantiation of Diagnostic Conversation Interface

Upstream requirements: RS_Diag_04166

⌈The Diagnostic Server instance shall provide as many instances of ara::diag::Conversation class ([SWS_DM_00693]) as the number of potential parallel Diagnostic Clients is configured by maxConversations.⌋

[SWS_DM_02003] Behavior of not configured SoftwareClusterDiagnosticDeploymentProps.maxConversations

Upstream requirements: RS_Diag_04166

⌈If the optional parameter SoftwareClusterDiagnosticDeploymentProps.maxConversations is not configured, the DM shall use a default value of '1' for that parameter.⌋

[SWS_DM_00841] Assignment of Diagnostic Conversation to Service Instances

Upstream requirements: RS_Diag_04166

⌈On establishment of a new `Diagnostic Conversation`, the `Diagnostic Server instance` shall assign this `Diagnostic Conversation` to an inactive `ara::diag::Conversation` class Instance, i.e. the field value of `ara::diag::ActivityStatusType` is set to `kInactive`. After assignment, the fields of the `ara::diag::Conversation` class Instance shall be updated according to the state of the given `Diagnostic Conversation`, i.e.,

- `ara::diag::ActivityStatusType` set to `kActive`,
- `ara::diag::Conversation::ConversationIdentifierType` matching the values of `apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` call, that initiated the creation of this `Diagnostic Conversation` (see [SWS_DM_00347]),
- a call to `ara::diag::Conversation::GetDiagnosticSession` will return the Diagnostic Session of this `Diagnostic Conversation`,
- a call to `ara::diag::Conversation::GetDiagnosticSecurityLevel` will return the Diagnostic Security Level of this `Diagnostic Conversation`.

⌋

[SWS_DM_00844] Updating DiagnosticConversation Service Instance fields

Upstream requirements: RS_Diag_04166

⌈During the life-cycle of a `Diagnostic Conversation`, the `Diagnostic Server instance` shall update the fields of the assigned `ara::diag::Conversation` class instance according to any change of the state of the `Diagnostic Conversation`.⌋

[SWS_DM_00843] Reset Service Instance fields on end of Diagnostic Conversation

Upstream requirements: RS_Diag_04166

⌈If the life-cycle of a `Diagnostic Conversation` ends, the `Diagnostic Server instance` shall reset

- the session reported with `ara::diag::Conversation::GetDiagnosticSession` to `kDefaultSession`
- the security level reported with `ara::diag::Conversation::GetDiagnosticSecurityLevel` to `kLocked`

of the assigned `ara::diag::Conversation` class instance.⌋

Besides the described informative character of the `ara::diag::Conversation` class Interface, it also provides methods for interaction with the state of a `Diagnostic Conversation`.

[SWS_DM_00842] Default session change trigger from AAs

Upstream requirements: RS_Diag_04006

⌈If `ara::diag::Conversation::ResetToDefaultSession` method is called, the `Diagnostic Server instance` shall complete the latest ongoing request and then switch the Diagnostic Session of this `Diagnostic Conversation` to Default Session.⌋

[SWS_DM_01355] Consecutive registration of notifier with SetActivityNotifier()

Upstream requirements: RS_Diag_04169

⌈In case of a consecutive call of `ara::diag::Conversation::SetActivityNotifier` of the corresponding `ara::diag::Conversation` instance, `DM` module shall overwrite the previous registered notifier.⌋

[SWS_DM_01356] Consecutive registration of notifier with SetDiagnosticSessionNotifier()

Upstream requirements: RS_Diag_04169, RS_Diag_04208

⌈In case of a consecutive call of `ara::diag::Conversation::SetDiagnosticSessionNotifier` of the corresponding `ara::diag::Conversation` instance, `DM` module shall overwrite the previous registered notifier.⌋

[SWS_DM_01357] Consecutive registration of notifier with SetSecurityLevelNotifier()

Upstream requirements: RS_Diag_04169, RS_Diag_04208

⌈In case of a consecutive call of `ara::diag::Conversation::SetSecurityLevelNotifier` of the corresponding `ara::diag::Conversation` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.2.2 Assignment of UDS requests to Diagnostic Conversations

A UDS request is always processed within the context of a `Diagnostic Conversation`. On reception, the `Diagnostic Server instance` has to choose from the following three options:

- assign the UDS request to an existing `Diagnostic Conversation`,
- establish a new `Diagnostic Conversation` and assign the UDS request to this `Diagnostic Conversation`,
- reject the UDS request.

The evaluation which option to choose involves several steps that are summarized in Figure 7.3. The following requirements provide the details.

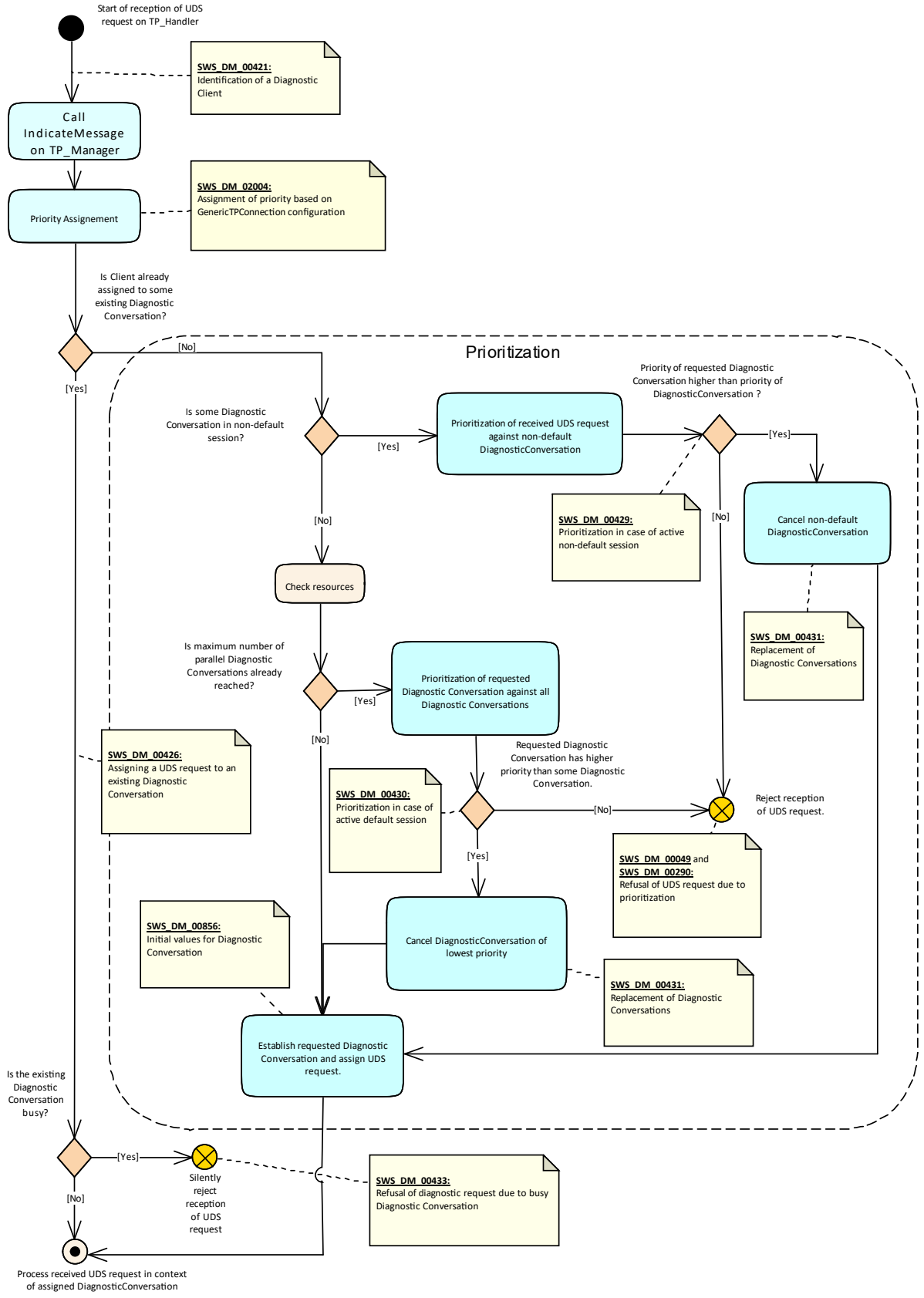

Figure 7.3: UDS request assignment to a Diagnostic Conversation and Prioritization

The Diagnostic Server instance handles a newly received UDS request as depicted in Figure 7.3.

[SWS_DM_00426] Assigning a UDS request to an existing Diagnostic Conversation

Upstream requirements: RS_Diag_04166

⌈If a UDS request is received and there already exists a Diagnostic Conversation associated to the transmitting Diagnostic Client, then the Diagnostic Server instance shall assign this UDS request to the same Diagnostic Conversation.⌋

Note that service 0x86 (RoE) serviceToRespondTo (STRT) messages may be sent to the respective Diagnostic Client by the Diagnostic Server instance at any time. In this case a reestablishment of an already ended DiagnosticConversation in default session has to be considered (see chapter 7.3.2.1.2).

[SWS_DM_01581] Assigning a UDS request to a new Diagnostic Conversation in active default session

Upstream requirements: RS_Diag_04166

⌈If a UDS request is received and there exists no Diagnostic Conversation associated to the transmitting Diagnostic Client, then the Diagnostic Server instance shall check the available Diagnostic Conversation resources according to [SWS_DM_00840]. In case a resource is free, the UDS request is assigned to the new Diagnostic Conversation. In case no resource is free, the priority handling according to [SWS_DM_00430] takes place.⌋

Note that the assignment of a UDS request to a Diagnostic Conversation does not necessarily mean that the UDS request is actually processed, see [SWS_DM_00433].

7.3.2.2.1 Prioritization

If the Diagnostic Server instance lacks resources for new Diagnostic Conversations, a prioritization of the requested Diagnostic Conversation against existing Diagnostic Conversations shall take place. For a Diagnostic Server instance , prioritization is required in case of an existing Diagnostic Conversation in non-default session.

[SWS_DM_02004] UDS request priority handling

Status: DRAFT
Upstream requirements: RS_Diag_04166

⌈The Diagnostic Server instance shall derive the priority of the diagnostic request from GenericTpConnection.priority where the source

address provided by apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage is in the range between GenericTpConnection.sourceAddressRangeStart and GenericTpConnection.sourceAddressRangeEnd.⌋

[SWS_DM_00428] Treatment of priority values

Upstream requirements: RS_Diag_04166

⌈The Diagnostic Server instance shall consider a lower value as higher priority and vice versa. In particular, priority value 0 represents highest priority.⌋

[SWS_DM_00429] Prioritization in active non-default session

Upstream requirements: RS_Diag_04166

⌈If a Diagnostic Conversation is in non-default session, the Diagnostic Server shall compare the priority of the requested Diagnostic Conversation against the priority of the given Diagnostic Conversation in non-default session. If the priority of the requested Diagnostic Conversation is higher than the priority of the Diagnostic Conversation in non-default Session, the Diagnostic Server instance shall replace the Diagnostic Conversation in non- default session by the requested Diagnostic Conversation according to [SWS_DM_00431] and assign the UDS request to the newly established Diagnostic Conversation.⌋

[SWS_DM_00430] Prioritization against all Diagnostic Conversations in active default session

Upstream requirements: RS_Diag_04166

⌈On prioritization, the Diagnostic Server instance shall compare the priority of the requested Diagnostic Conversation against the priorities of the existing Diagnostic Conversations:

- If all priorities of the existing Diagnostic Conversations are higher or equal to the priority of the requested Diagnostic Conversation, the Diagnostic Server instance shall refuse the UDS request according to [SWS_DM_00049] and [SWS_DM_00290].
- If some priority of the existing Diagnostic Conversations is lower than the priority of the requested Diagnostic Conversation, the Diagnostic Server instance shall replace the Diagnostic Conversation of lowest priority by the requested Diagnostic Conversation according to [SWS_DM_00431] and assign the UDS request to the newly established Diagnostic Conversation.

⌋

7.3.2.2.2 Replacement of Diagnostic Conversations and initial values

[SWS_DM_00431] Replacement of Diagnostic Conversations

Upstream requirements: RS_Diag_04167

⌈On replacement of a given `Diagnostic Conversation` by a requested `Diagnostic Conversation`, the `Diagnostic Server instance` shall cancel the given `Diagnostic Conversation` according to [SWS_DM_00277], [SWS_DM_00278], [SWS_DM_00279], [SWS_DM_00280], [SWS_DM_00847] and establish a new `Diagnostic Conversation` as requested.⌋

[SWS_DM_00856] Initial values for Diagnostic Conversation

Upstream requirements: RS_Diag_04166

⌈For a newly established Diagnostic Conversation, the `Diagnostic Server instance` shall use the following initial values:

- Session set to Default Session (`kDefaultSession`), which is synonymous with returning an according `ara::diag::SessionControlType` when `ara::diag::Conversation::GetDiagnosticSession` is called and
- Security Level set to status Locked (`kLocked`), which is synonymous with returning an according `ara::diag::SecurityLevelType` when `ara::diag::Conversation::GetDiagnosticSecurityLevel` is called .

⌋

7.3.2.2.3 Refusal of incoming diagnostic request

[SWS_DM_00433] Refusal of diagnostic request due to busy Diagnostic Conversation

Upstream requirements: RS_Diag_04020

⌈If a UDS request is assigned to a `Diagnostic Conversation` that has not finished processing of a formerly assigned UDS request, then the `Diagnostic Server instance` shall ignore the new UDS request according to [SWS_DM_00386].⌋

[SWS_DM_00049] Refusal of diagnostic request due to prioritization with BusyRepeatRequest

Upstream requirements: RS_Diag_04167

⌈If prioritization demands refusal of an incoming UDS request and the configuration parameter `DiagnosticCommonProps.responseOnSecondDeclinedRequest` is TRUE, the `Diagnostic Server instance` shall accept this request according to

[SWS_DM_00385] without further processing and a negative response with NRC 0x21 (BusyRepeatRequest) shall be issued for this request.⌋

[SWS_DM_00290] Refusal of diagnostic request due to prioritization without response

Upstream requirements: RS_Diag_04167

⌈If prioritization demands refusal of an incoming UDS request and the configuration parameter `DiagnosticCommonProps.responseOnSecondDeclinedRequest` is FALSE, the `Diagnostic Server instance` shall ignore this request according to [SWS_DM_00386] without further processing and no response shall be issued.⌋

7.3.2.3 Handling Authentication State and DynamicAccessLists

This chapter describes the interfaces available to the application for handling of the Authentication States of the Diagnostic Clients and their corresponding DynamicAccessLists. The parts specified in this chapter are independent of the parts specified for the UDS Service Authentication (0x29), and may be used also with custom methods for authentication of clients.

7.3.2.3.1 ExternalAuthentication

In AUTOSAR Adaptive, a major part of the client authentication process is handled in the Application. It is therefore necessary for the application to convey the Authentication state to the `Diagnostic Server instance` of the `DM`. Since the `Diagnostic Server instance` must handle the Authentication States and Roles independently for each `Diagnostic Client`, the Application must first receive a ClientAuthentication instance from the `DM`. This can be done using the class `ara::diag::ExternalAuthentication`.

With the DiagnosticExternalAuthenticationIdentification model element a range or single source Addresses of diagnostic clients can be defined. The number of available DiagnosticExternalAuthenticationIdentification elements define the number of instances of the `ara::diag::ClientAuthentication` class. The intention of this element is to allow the application an authentication for a specific `diagnostic client` with a fixed `source address` or with a range of source addresses. The range of source addresses is used if the final `source address` of the client is within a range and not known upfront (during compile time).

[SWS_DM_01202] Get ClientAuthentication Instance

Upstream requirements: RS_Diag_04240

⌈If the application calls any of the overloaded methods `ara::diag::ExternalAuthentication::Get` of the class `ara::diag::ExternalAuthentication`, the

Diagnostic Manager shall return an instance of the ara::diag::ClientAuthentication class that is handling the Authentication State of the passed metaInfo or Diagnostic Client Address.⌋

[SWS_DM_01203] GetAll ClientAuthentication Instance

Upstream requirements: RS_Diag_04240

⌈If the application calls the method ara::diag::ExternalAuthentication::GetAll of the class ara::diag::ExternalAuthentication, the Diagnostic Server instance shall return all the existing instances of the ara::diag::ClientAuthentication class.⌋

7.3.2.3.2 ClientAuthentication

Once the Application has received an instance of the ara::diag::ClientAuthentication class, it may pass the Authentication State and Authentication Roles to the Diagnostic Manager using this instance. The Diagnostic Server instance maintains the Authentication State and Authentication Role for each Diagnostic Client.

[SWS_DM_01229] Support for authentication per Diagnostic Client

Upstream requirements: RS_Diag_04251

⌈The Diagnostic Server instance shall support the Authentication service independently for every Diagnostic Client.⌋

NOTE : The authentication status on one Diagnostic Client shall not influence the access restrictions on a different Diagnostic Connection.

[SWS_DM_01204] Default Authentication Role

Upstream requirements: RS_Diag_04240

⌈The DiagnosticAuthRole(s) with attribute “isDefault” set to TRUE shall be considered by the Diagnostic Server instance as the Default Authentication Role(s) of clients in the Authentication State kDeAuthenticated.⌋

[SWS_DM_01205] Default Authentication State

Upstream requirements: RS_Diag_04240

⌈On startup, the default Authentication state for a client shall be ‘kDeAuthenticated’.⌋

[SWS_DM_01206] Set AuthenticationRole

Upstream requirements: RS_Diag_04240

⌈If the application calls the `ara::diag::ClientAuthentication::Authenticate` method, the `Diagnostic Server instance` shall set the Authentication State of the `ara::diag::ClientAuthentication` instance to `kAuthenticated` and the Authentication Role of the `ara::diag::ClientAuthentication` instance to the user role(s) that are passed in the method call. The Diagnostic Server Instance shall return an instance of the `ara::diag::ClientAuthenticationHandle` to the application.⌋

[SWS_DM_01570] Lifetime of overridden default roles

Upstream requirements: RS_Diag_04240

⌈If `ara::diag::ClientAuthentication::Authenticate`() is called and the `Diagnostic Server instance` switches to `kAuthenticated` State, the overridden default roles shall be reset to `DiagnosticAuthRole.isDefault`.⌋

[SWS_DM_01207] Get Authentication State

Upstream requirements: RS_Diag_04240

⌈If the application calls the `ara::diag::ClientAuthentication::GetState` method of the class `ara::diag::ClientAuthentication`, the `Diagnostic Server instance` shall return the current Authentication State of the Client.⌋

[SWS_DM_01208] Authentication State Change Notifier

Upstream requirements: RS_Diag_04240

⌈If the application calls the method `ara::diag::ClientAuthentication::SetNotifier` of the class `ara::diag::ClientAuthentication`, the `Diagnostic Server instance` shall call the passed notifier-function whenever there is a change in the Authentication State of the client.⌋

[SWS_DM_01209] Temporarily change Default Roles

Upstream requirements: RS_Diag_04239

⌈If the client is in `kDeAuthenticated` state and the application calls the method `ara::diag::ClientAuthentication::OverrideDefaultRoles` of the class `ara::diag::ClientAuthentication`, the `Diagnostic Server instance` shall change the default roles of the client to the passed 'defaultRoles' for a time period passed in the parameter 'timeout'. On successful execution of this method, the `Diagnostic Server instance` shall return an instance of the `ara::diag::ClientAuthenticationHandle` to the application.⌋

[SWS_DM_01210] DeAuthenticate due to client inactivity

Upstream requirements: RS_Diag_04240

⌈If a client is in default session and in state `kAuthenticated`, the `Diagnostic Server instance` shall set the Authentication State of the client to `kDeAuthenticated` if after the last `apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation` from this client no further request was received from the same client for a time of `authenticationTimeout`.⌋

[SWS_DM_01211] Transition to DeAuthenticated state on S3server timeout

Upstream requirements: RS_Diag_04240

⌈On an S3 Server timeout, the `Diagnostic Server instance` shall reset the Authentication State to `kDeAuthenticated` for the client which timed out.⌋

[SWS_DM_01212] Transition from Authenticated to DeAuthenticated State

Upstream requirements: RS_Diag_04240

⌈If the Authentication State of a Client changes from `kAuthenticated` to `kDeAuthenticated`, the `Diagnostic Server instance` shall

- set the Authentication Role of the Client to the default roles as defined in [SWS_DM_01204]
- clear all DynamicAccessList entries associated with the Client

⌋

[SWS_DM_01360] Consecutive registration of notifier with ClientAuthentication::SetNotifier()

Upstream requirements: RS_Diag_04251

⌈In case of a consecutive call of `ara::diag::ClientAuthentication::SetNotifier` of the corresponding `ara::diag::ClientAuthentication` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.2.3.3 ClientAuthenticationHandle

A `ara::diag::ClientAuthenticationHandle` instance is provided to the application by the `Diagnostic Server instance`, when either an `ara::diag::ClientAuthentication::OverrideDefaultRoles` method or an `ara::diag::ClientAuthentication::Authenticate` method is successfully completed.

The Diagnostic Manager maintains a “DynamicAccessList” for every client that is authenticated. The DynamicAccessList may provide additional access of Diagnostic Resources to an authenticated client apart from the configurations described in the Diagnostic Extract. The DynamicAccessList and the Authentication Status may be controlled by the application using the `ara::diag::ClientAuthenticationHandle`.

[SWS_DM_01213] Set DynamicAccessList

Upstream requirements: RS_Diag_04240

⌈If the application calls the `ara::diag::ClientAuthenticationHandle::Set` method of the class `ara::diag::ClientAuthenticationHandle`, the `Diagnostic Server instance` shall replace the DynamicAccessList of the authenticated client with the DynamicAccessList passed by the application.⌋

[SWS_DM_01215] Extend the DynamicAccessList

Upstream requirements: RS_Diag_04240

⌈If the application calls the `ara::diag::ClientAuthenticationHandle::Append` method of the class `ara::diag::ClientAuthenticationHandle`, the `Diagnostic Server instance` shall extend the DynamicAccessList of the authenticated client with the DynamicAccessList passed by the application.⌋

[SWS_DM_01216] Revoke an authentication

Upstream requirements: RS_Diag_04240

⌈If the application calls the `ara::diag::ClientAuthenticationHandle::Revoke` method of the class `ara::diag::ClientAuthenticationHandle`, the `Diagnostic Server instance` shall set the Authentication State of the client to ‘`kDeAuthenticated`’.⌋

[SWS_DM_01217] Refresh timeouts

Upstream requirements: RS_Diag_04240

⌈If an application calls the method `ara::diag::ClientAuthenticationHandle::Refresh` of the class `ara::diag::ClientAuthenticationHandle`, the `Diagnostic Server instance` shall perform one of the following:

- If the `ara::diag::ClientAuthenticationHandle` was returned by the `ara::diag::ClientAuthentication::OverrideDefaultRoles` method, refresh the time period for which the OverrideDefaultRoles is valid. Refer [SWS_DM_01209].
- If the `ara::diag::ClientAuthenticationHandle` was returned by the `ara::diag::ClientAuthentication::Authenticate` method, refresh the client inactivity time period. Refer [SWS_DM_01210].

⌋

7.3.2.3.4 DynamicAccessList Creation and Update

The Diagnostic Manager provides an interface to allow the application to build a DynamicAccessList. The DynamicAccessList is a series of diagnostic request patterns that provide additional access of diagnostic resources to an authenticated client. The DynamicAccessList may be created by the application using the C++ methods described in this chapter.

It is the general idea of AUTOSAR to have a common certificate layout, that is applicable for all ECUs (classic and adaptive), which is defined by [RS_Diag_04234] While CP has means to specify the whitelist layout in adaptive AUTOSAR this task dedicated to an application and the DM has no control of it. But still it is recommended to use the whitelist layout which is defined in CP DCM.

[SWS_DM_01214] Default DynamicAccessList

Upstream requirements: RS_Diag_04240

⌈On startup, the DynamicAccessList of all clients shall be empty.⌋

[SWS_DM_01218] Building a new DynamicAccessList

Upstream requirements: RS_Diag_04240, RS_Diag_04234

⌈If any of the overloads of the method `ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder` of the class `ara::diag::DiagnosticServiceDynamicAccessList` is called by the application, the `Diagnostic Server instance` shall create a new DynamicAccessList beginning with the single-byte or byte-string pattern passed by the application. The `Diagnostic Server instance` shall return an instance of the class `ara::diag::DynamicAccessListDiagServiceBuilder` to the application.⌋

[SWS_DM_01219] Adding patterns to a DynamicAccessList

Upstream requirements: RS_Diag_04240, RS_Diag_04234

⌈If any of the overloads of the method `ara::diag::DynamicAccessListDiagServiceBuilder::Add` of the class `ara::diag::DynamicAccessListDiagServiceBuilder` are called by the application, the `Diagnostic Server instance` shall add the requested pattern to the DynamicAccessList of the client and return an instance of the same `ara::diag::DynamicAccessListDiagServiceBuilder` object to the application.⌋

The returned instance of the same object may be used by the application to further add patterns to the DynamicAccessList.

[SWS_DM_01220] Adding wildcards to a DynamicAccessList

Upstream requirements: RS_Diag_04240, RS_Diag_04234

⌈If the method `ara::diag::DynamicAccessListDiagServiceBuilder::Any` of the class `ara::diag::DynamicAccessListDiagServiceBuilder` is called by the application, the `Diagnostic Server instance` shall add the passed number of bytes to the DynamicAccessList, but shall not consider them during pattern matching.⌋

For Example, 22F1XX could be used in the pattern to add all ReadDataByIdentifier Requests for DIDs beginning with 0xF1 to the DynamicAccessList.

[SWS_DM_01221] End patterns of a DynamicAccessList

Upstream requirements: RS_Diag_04240, RS_Diag_04234

⌈If any of the overloads of the method `ara::diag::DynamicAccessListDiagServiceBuilder::EndsWith` of the class `ara::diag::DynamicAccessListDiagServiceBuilder` are called by the application, the `Diagnostic Server instance` shall add the requested pattern to the end of the DynamicAccessList.⌋

[SWS_DM_01222] Finalize a DynamicAccessList

Upstream requirements: RS_Diag_04240, RS_Diag_04234

⌈If the method `ara::diag::DynamicAccessListDiagServiceBuilder::Build` of the class `ara::diag::DynamicAccessListDiagServiceBuilder` is called by the application, the `Diagnostic Server instance` shall finalize the DynamicAccessList.⌋

After successful execution of this method, the application may use the created DynamicAccessList as described in [SWS_DM_01213].

7.3.2.4 Diagnostic Service Authentication checks

The UDS service Authentication (0x29) is used by the `ECU` as a means to identify the client and provide the relevant access to diagnostic resources, based on the client's 'role'. Depending on the authenticated role and the access-list, a dynamic set of diagnostic services is available to the client. The `Diagnostic Server instance` verifies if a received diagnostic service is accessible to the client or not.

[SWS_DM_01739] Authentication disabled

Upstream requirements: RS_Diag_04233

⌈If `DiagnosticAccessPermission.authenticationEnabled` does not exist, no authentication checks shall be performed.⌋

[SWS_DM_01223] Diagnostic service role verification

Upstream requirements: RS_Diag_04233

⌈If `DiagnosticAccessPermission.authenticationEnabled` and `DiagnosticAuthRoleProxy.authenticationRole` exist, then The `Diagnostic Server instance` shall check if a diagnostic service execution is permitted in the current Authentication State and Authentication Role (Refer [SWS_DM_01206]). The roles that are allowed to execute the diagnostic services are configured with the parameter `DiagnosticAccessPermission.DiagnosticAuthRoleProxy.authenticationRole`. The `Diagnostic Server instance` shall perform the following checks in the order given below. If a check grants access to a service, the remaining checks are skipped and the service is processed by the `Diagnostic Server instance`.

- Checks on service ID level
  - this is skipped for services with identifiers (DID / RID)
  - this is skipped if this service has subfunctions and none of these subfunctions grants access in the current authenticated role
- Checks on service ID and sub-function level
- Checks for services with one or multiple DIDs
- Check on dynamically defined DIDs
- Checks on service 0x31 per sub-function
- Checks on service 0x19 parameter MemorySelection
- Checks on service 0x14 parameter MemorySelection

⌋

Note: Please note that the protection of the fault memory is handled different for primary and user defined fault memory. While authentication for the primary fault memory is realized as a protection of service ID and subfunction level, the user defined fault memory is protected via the memorySelection parameter from ISO 14229-1[1]. This seems strange at first view, but it matches the use case of how user defined memories are used. Protecting user defined memory is mainly driven by security events or ECU supplier protecting their data. Having valid access rights for a user defined memory will than give full access to that user defined memory, including ClearDiagnosticInformation and reading all data of that user defined memory.

[SWS_DM_01224] Diagnostic service dynamic access-rights verification

Upstream requirements: RS_Diag_04232

⌈If the check in [SWS_DM_01223] is unsuccessful, the `Diagnostic Server instance` shall additionally check if the requested Diagnostic Service is allowed by the client DynamicAccessList by applying a pattern match on the received `UDS` request

towards the DynamicAccessList elements. The check is considered as successful if all the bytes of one entry of the DynamicAccessList are matching the UDS request.⌋

Further bytes in the UDS request are not relevant.

Example 1:

DynamicAccessList: 31 01 13F4

Any Routine Control Service with StartRoutine 13F4 would be accepted, regardless of the routineControlOptionRecords.

Example 2:

DynamicAccessList: 11

Any ECUReset Service would be accepted, regardless of the subfunction.

[SWS_DM_01225] Response behavior of services without access rights

Upstream requirements: RS_Diag_04232

⌈If the service execution verification fails due to a failed check in scope of [SWS_DM_01223] and [SWS_DM_01224], the Diagnostic Server instance shall send a negative Response with NRC '0x34 (authenticationRequired)' and stop the service processing.⌋

[SWS_DM_01376] Response behavior of SOVD services without access rights

Upstream requirements: RS_Diag_04268

⌈If the request is an SOVD request and the service execution verification fails due to a failed check in scope of [SWS_DM_01223], the Diagnostic Server instance shall send a HTTP response status code 401 (Unauthorized) and error_code set to insufficient-access-rights and stop the service processing.⌋

7.3.2.5 UDS request Validation/Verification

[SWS_DM_00096] Validation Steps and Order

Upstream requirements: RS_Diag_04196, RS_Diag_04203

⌈The Diagnostic Server instance shall execute the request validation, negative response code determination and processing according to ISO 14229-1[1].⌋

ISO 14229-1[1] describes a common processing for all requests in "Figure 5 – General server response behavior". There are further optional SID specific processing sequences. This document describes the Diagnostic Server instance behavior for certain types of checks:

- **manufacturer specific failure detected?** Decision by applying manufacturer specific checks according to section 7.3.2.5.4
- **SID supported?** Decision according to section 7.3.2.5.2
- **SID supported in active session?** Decision according to section 7.3.2.5.3
- **SID security check o.k.?** Decision according to section 7.3.2.5.3
- **supplier-specific failure detected?** Decision by applying supplier-specific checks according to section 7.3.2.5.4

[SWS_DM_00097] Abort on failed verification step

Upstream requirements: RS_Diag_04196

⌈Whenever one of the verification steps fails, further processing of the request shall be aborted and a negative response shall be sent back.⌋

The negative response code to be used will be defined in each step described in the following sections.

7.3.2.5.1 UDS request format checks

[SWS_DM_00098] UDS message checks

Upstream requirements: RS_Diag_04203, RS_Diag_04228

⌈The `Diagnostic Server instance` shall check, whether the diagnostic request is syntactically correct. I.e. whether it conforms to ISO 14229-1 message format specification. If it does not conform, the Verification shall be considered as failed and the negative response code shall be 0x13 (`incorrectMessageLengthOrInvalidFormat`).⌋

7.3.2.5.2 Supported service checks

[SWS_DM_00099] Supported Service SID level checks

Upstream requirements: RS_Diag_04203

⌈The `Diagnostic Server instance` shall check, whether there is a configured internal or external service processor for the incoming diagnostic request. If there is no service processor on `SID` level, the Verification shall be considered as failed and the negative response code shall be 0x11 (`serviceNotSupported`).⌋

[SWS_DM_00100] Supported Service subfunction level checks

Upstream requirements: RS_Diag_04203

⌈The Diagnostic Server instance shall check, whether there is a configured internal or external service processor for the incoming diagnostic request. If there exists a service processor on SID level, but not for the subfunction of the request, the Verification shall be considered as failed and the negative response code shall be 0x12 (subFunctionNotSupported).⌋

7.3.2.5.3 Session and Security Checks

[SWS_DM_00101] Session Access SID level Permission

Upstream requirements: RS_Diag_04203, RS_Diag_04006, RS_Diag_04226

⌈The Diagnostic Server instance shall check, whether the service processor (DiagnosticServiceInstance), which is assigned to handle the service has the permission to process the service in the current Diagnostic Session according to its DiagnosticAccessPermission.diagnosticSession. If DiagnosticServiceInstance has no access permissions in the current Diagnostic Session and:

- either the SID of the service has no subfunction
- or all other sub-functions also have no access permissions in the current Diagnostic Session,
- service has no identifier (DID / RID) or all other identifers have no access permission in the current session.

the Verification shall be considered as failed and the negative response code shall be 0x7F (serviceNotSupportedInActiveSession).⌋

[SWS_DM_00102] Session Access subfunction level Permission

Upstream requirements: RS_Diag_04203, RS_Diag_04006, RS_Diag_04226

⌈The Diagnostic Server instance shall check, whether the service processor (DiagnosticServiceInstance), which is assigned to handle the service has the permission to process the service in the current Diagnostic Session according to its DiagnosticAccessPermission.diagnosticSession. If DiagnosticServiceInstance has no access permissions in the current Diagnostic Session and:

- the SID of the service has subfunctions
- and at least one other sub-functions has access permissions in the current Diagnostic Session,

the Verification shall be considered as failed and the negative response code shall be 0x7E (subFunctionNotSupportedInActiveSession).⌋

[SWS_DM_00103] Security Access level Permission

Upstream requirements: RS_Diag_04203, RS_Diag_04005, RS_Diag_04227

⌈The Diagnostic Server instance shall check, whether the service processor (DiagnosticServiceInstance), which is assigned to handle the service has the permission to process the service in the current Security-Level according to its DiagnosticAccessPermission.securityLevel. If DiagnosticServiceInstance has no access permissions in the current Security-Level, the Verification shall be considered as failed and the negative response code shall be 0x33 (securityAccessDenied).⌋

[SWS_DM_00450] Security Access subfunction level Permission

Upstream requirements: RS_Diag_04203, RS_Diag_04227

⌈The Diagnostic Server instance shall check, whether the service processor (DiagnosticServiceInstance), which is assigned to handle the service has the permission to process the service in the current Security Level according to its DiagnosticAccessPermission.securityLevel. If DiagnosticServiceInstance has no access permissions in the current Security Level and:

- the SID of the service has subfunctions
- and at least one other sub-functions has access permissions in the current Security Level,

the Verification shall be considered as failed and the negative response code shall be 0x33 (securityAccessDenied).⌋

[SWS_DM_01951] No session checks if no diagnostic session inside the diagnostic access permission

Upstream requirements: RS_Diag_04226

⌈If the Diagnostic Server instance is evaluating the diagnostic session to execute the current diagnostic service and no DiagnosticAccessPermission or DiagnosticAccessPermission.diagnosticSession references are configured, the Diagnostic Service instance shall directly execute the diagnostic service and skip the session checks.⌋

[SWS_DM_01952] No security level checks if no securityLevel inside the diagnostic access permission

Upstream requirements: RS_Diag_04227

⌈If the Diagnostic Server instance is evaluating the security level to execute the current diagnostic service and no DiagnosticAccessPermission or DiagnosticAccessPermission.securityLevel references are configured, the Diagnostic Service instance shall directly execute the diagnostic service and skip the security level checks.⌋

7.3.2.5.4 Manufacturer and Supplier Permission Checks and Confirmation

To allow manufacturer specific UDS service pre-processing or filtering, ISO 14229-1[1] defines manufacturer and supplier specific callouts. There are various use cases for these callouts, among them are:

- UDS message filtering
- Adding project or customer specific diagnostic service processing

UDS and the DM allows multiple of these callouts and the result of each of the callout can be one of:

- Continue to process the service by the DM
- Discarding the received diagnostic message without further response
- Forcing a certain NRC to be send

Both, manufacturer and suppliers specific service callouts, are realized by ara::diag::ServiceValidation::Validate according to its modeled instance given by [TPS_MANI_01352] and [constr_10063].

[SWS_DM_01252] Support of manufacturer service validations

Upstream requirements: RS_Diag_04199

⌈The DM shall support manufacturer specific service validation checks according to ISO 14229-1[1] by calling ara::diag::ServiceValidation::Validate for each configured manufacturer service check.⌋

[SWS_DM_01253] Support of supplier service validations

Upstream requirements: RS_Diag_04199

⌈The DM shall support supplier specific service validation checks according to ISO 14229-1[1] by calling ara::diag::ServiceValidation::Validate for each configured supplier service check.⌋

[SWS_DM_01254] Continue service processing after validation

Upstream requirements: RS_Diag_04199

⌈If a call to ara::diag::ServiceValidation::Validate returns without an error, the Diagnostic Server instance shall continue to process the service according to ISO 14229-1[1].⌋

[SWS_DM_01255] NRC after failed service validation

Upstream requirements: RS_Diag_04199

⌈If a call to ara::diag::ServiceValidation::Validate returns any error code except kNoProcessingNoResponse, the validation is be considered as failed and

a negative response code equal to the value of the error code according to ara::diag::DiagUdsNrcErrc shall be sent as response.⌋

[SWS_DM_00859] Confirmation of service processing

Upstream requirements: RS_Diag_04019, RS_Diag_04172

⌈The Diagnostic Server instance shall call the method ara::diag::ServiceValidation::Confirmation on every service instances for which ara::diag::ServiceValidation::Validate was called. If message handling results in sending a positive or negative response, the ara::diag::ServiceValidation::Confirmation call shall be deferred after reception of apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation. In any other case, it shall be the last step of request processing.⌋

[SWS_DM_00860] No service processing

Upstream requirements: RS_Diag_04196, RS_Diag_04199

⌈If any ara::diag::ServiceValidation::Validate returns kNoProcessingNoResponse, the Diagnostic Server instance shall discard the message of received diagnostic request.⌋

7.3.2.5.5 Condition checks

In some cases, diagnostic functionality shall only be executed if the vehicle is in a certain state. An example is the condition that the vehicle is stopped (vehicle speed equals 0).

[SWS_DM_00111] Configurable environment condition checks

Upstream requirements: RS_Diag_04199

⌈The Diagnostic Server instance shall perform a condition check when the ISO 14229-1[1] mentions a service specific "Condition check" in the defined NRC handling for a given diagnostic service. The Diagnostic Server instance shall send the configured NRC value (see [SWS_DM_00289]) if the condition is not fulfilled.⌋

[SWS_DM_00112] Condition check definition

Upstream requirements: RS_Diag_04199

⌈The Diagnostic Server instance shall execute a condition check according to [SWS_DM_00111] by the presence of a DiagnosticEnvironmentalCondition referenced in the role environmentalCondition by the processed DiagnosticServiceInstance.⌋

[SWS_DM_00286] Configurable environmental condition check execution

Upstream requirements: RS_Diag_04199

⌈The Diagnostic Server instance shall execute an environmental condition check before executing the requested service if defined. (see DiagnosticEnvironmentalCondition element from DEXT [3]).⌋

[SWS_DM_00287] Configurable environmental condition check criteria

Upstream requirements: RS_Diag_04199

⌈The environmental condition check shall be done by evaluation of the configured DiagnosticEnvConditionFormula.⌋

The DiagnosticEnvConditionFormula may reference a DiagnosticDataElement by a DiagnosticEnvDataCondition with a logical operator given as DiagnosticEnvCompareCondition.

[SWS_DM_00288] Configurable environmental condition check evaluates to `TRUE`

Upstream requirements: RS_Diag_04199

⌈If the computation of the DiagnosticEnvConditionFormula evaluated to TRUE, the Diagnostic Server instance shall execute the requested service.⌋

[SWS_DM_00970] Behavior of failed data element retrieval

Upstream requirements: RS_Diag_04199

⌈If the retrieval of the dataElement failed due to an external processor has an error of ara::diag::DiagUdsNrcErrorDomain, the DM shall treat the DiagnosticEnvConditionFormulaPart as condition not fulfilled and trigger a Log and Trace message.⌋

[SWS_DM_00289] Configurable environmental condition check evaluates to `FALSE`

Upstream requirements: RS_Diag_04199

⌈The Diagnostic Server instance shall send the NRC defined in nrcValue, if the computation of the DiagnosticEnvConditionFormula evaluated to FALSE. If nrcValue does not define a NRC, the Diagnostic Server instance shall send NRC 0x22 (ConditionsNotCorrect).⌋

7.3.2.6 UDS response handling

[SWS_DM_01258] Response handling

Upstream requirements: RS_Diag_04196, RS_Diag_04020, RS_Diag_04249

⌈The Diagnostic Server instance shall process diagnostic responses according to ISO 14229-1[1]. This includes sending of positive and negative responses, as well as suppression of negative or positive responses.⌋

[SWS_DM_00368] DM takes care of Response Pending Messages

Upstream requirements: RS_Diag_04016, RS_Diag_04249

⌈If the processing of a diagnostic service requires more time than allowed by the P2/P2* timer of the current session, the Diagnostic Server instance shall send a negative response with NRC 0x78 (requestCorrectlyReceivedResponsePending) according to ISO 14229-1[1].⌋

[SWS_DM_00369] Maximum number of busy responses

Upstream requirements: RS_Diag_04016, RS_Diag_04249

⌈If the number of negative responses for a requested diagnostic request reaches the value defined in the configuration parameter maxNumberOfRequestCorrectlyReceivedResponsePending, the Diagnostic Server instance module shall cancel the processing of the active diagnostic internal or external request processing, according to [SWS_DM_00277], [SWS_DM_00278] and send a negative response with NRC 0x10 (generalReject).⌋

[SWS_DM_01257] ResourceTemporarilyNotAvailable NRC handling

Upstream requirements: RS_Diag_04016, RS_Diag_04249

⌈If the DM is processing a diagnostic service using a ara::diag interface different than ara::diag::ServiceValidation, that requires the offer(), and if this interface is not offered yet or not offered any more, the DM shall return an NRC 0x94 (ResourceTemporarilyNotAvailable).⌋

7.3.2.6.1 NRCs created by application

Some UDS services require external service processing. If such a diagnostic service is received, the DM will call the corresponding ara::diag service processing methods. The service processing in the application can result in positive response with valid response data but also in a NRC where the NRC code is provided by the application.

If the external service processor cannot proces the diagnostic service and requests to send a NRC as response it uses the error code in the enum class DiagUdsNrcErrc as return value. This enum class is defined in [SWS_DM_00526] and contains a predefined set of NRC codes that are standardized in ISO 14229-1[12]. An application may choose any of the values within [SWS_DM_00526] to trigger the corresponding NRC as result of the serivce processing. In some rare cases the application will trigger NRC codes that are not standardized in ISO 14229-1[12]. These NRCs are in ranges that are or marked as manufacturer specific or reserved. It is explicitely allowed that an application can use any value in the interval 0x01 to 0xFE as NRC response. While not the entire range between 0x01 and 0xFE is covered by value defintions in [SWS_DM_00526], the application can use a static cast to create such an NRC code. Example how to create the NRC 0xF0:

```
static_cast<DiagUdsNrcErrc>(0xF0)
```

The DM itself will accept any value in the interval 0x01 to 0xFE of the returned value from application. This also means, that any NRC code can be returned by any ara::diag method with enum class DiagUdsNrcErrc. The DM will not make any service specific assumption that certain NRCs are not possible with a certain ara::diag method. Of cause not all 254 NRC codes make sense for each diagnostic services. Some NRCs are only needed for a subset of diagnostic services (e.g NRC 0x24 is not used for service ReadDataByIdentifier).

[SWS_DM_02059] Derive NRC from DiagUdsNrcErrc

Upstream requirements: RS_Diag_04196

⌈The DM shall derive the UDS NRC from the DiagUdsNrcErrc error code returned by the application (e.g. by doing a `static_cast` into a 1 byte NRC value). Valid values are only in the range 0x01 to 0xFE.⌋

[SWS_DM_02060] Reaction on ApplicationError

Upstream requirements: RS_Diag_04196

⌈If the return value of an ara::diag interface has an error of ara::diag::DiagUdsNrcErrorDomain, the Diagnostic Server instance shall return a negative response with the value of that error code.⌋

7.3.2.7 Keep track of active non-default sessions

[SWS_DM_00380] Support for S3 timer

Upstream requirements: RS_Diag_04006, RS_Diag_04249

⌈The Diagnostic Server instance shall provide support for $S3_{Server}$ (session timeout) with a server specific value according to [SWS_DM_01747]. The timer handling shall be implemented according to ISO 14229-2[12].⌋

[SWS_DM_01747] S3 timer value

Upstream requirements: RS_Diag_04249, RS_Diag_04006

⌈If the parameter DiagnosticSessionControlClass.s3ServerTimeout is not configured, the Diagnostic Server instance (Software Cluster), shall take the value of 5000ms as the S3Server timeout. If the parameter DiagnosticSessionControlClass.s3ServerTimeout is configured, the Diagnostic Server Instance (Software Cluster) shall take the configured value as the S3Server timeout.⌋

[SWS_DM_00381] Session timeout

Upstream requirements: RS_Diag_04006, RS_Diag_04249

⌈Whenever a non-default session is active and when the session timeout ($S3_{Server}$) is reached without receiving any diagnostic request, the Diagnostic Server instance shall reset to the default session state. Diagnostic Server instance internal states for service processing shall be reset according to ISO 14229-2[12].⌋

[SWS_DM_00382] Session timeout start

Upstream requirements: RS_Diag_04006, RS_Diag_04249

⌈The session timeout timer ($S3_{server}$) shall be started on

- Completion of any final response message or an error indication during sending of the response ([SWS_DM_00312])
- Completion of the requested action in case no response message (positive and negative) is required / allowed.
- In case of an error during the reception of a multi-frame request message ([SWS_DM_00310])

Start of $S3_{Server}$ means reset the timer and start counting from the beginning.⌋

[SWS_DM_00383] Session timeout stop

Upstream requirements: RS_Diag_04006, RS_Diag_04249

⌈The session timeout timer ($S3_{Server}$) shall be stopped when the reception of an UDS message was indicated ([SWS_DM_00309]).⌋

[SWS_DM_00812] Re-enabling on transition to default session

Upstream requirements: RS_Diag_04006, RS_Diag_04249

⌈If DTC setting is disabled and DM is transitioning into default session, then DM shall enable the DTC setting again.⌋

7.3.2.8 UDS service processing

This chapter describes the UDS service processing behavior of the Diagnostic Server instance.

[SWS_DM_00127] Availability of diagnostic service processors

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall provide a service processor on SID level for all services by existence of a DiagnosticServiceClass referenced by a DiagnosticServiceInstance.serviceClass.⌋

7.3.2.8.1 Supported UDS Services

The Diagnostic Server instance shall support the following listed UDS services:

SID	Service	Support Type	Reference
0x10	DiagnosticSessionControl	Internally	7.3.2.8.4
0x11	ECUReset	Externally	7.3.2.8.5
0x14	ClearDiagnosticInformation	Internally	7.3.2.8.8
0x19	ReadDTCInformation	Internally	7.3.2.8.9
0x22	ReadDataByIdentifier	Internally & Externally	7.3.2.8.10
0x27	SecurityAccess	Internally & Externally	7.3.2.8.11
0x28	CommunicationControl	Externally	7.3.2.8.12
0x29	Authentication	Externally	7.3.2.8.13
0x2A	ReadDataByPeriodicIdentifier	Internally	7.3.2.8.14
0x2C	DynamicallyDefineDataIdentifier	Internally	7.3.2.8.15
0x2E	WriteDataByIdentifier	Externally	7.3.2.8.16
0x31	RoutineControl	Externally	7.3.2.8.17
0x34	RequestDownload	Externally	7.3.2.8.18
0x35	RequestUpload	Externally	7.3.2.8.19
0x36	TransferData	Externally	7.3.2.8.20
0x37	RequestTransferExit	Externally	7.3.2.8.21
0x38	RequestFileTransfer	Externally	7.3.2.8.22
0x3E	TesterPresent	Internally	7.3.2.8.23
0x85	ControlDTCSetting	Internally	7.3.2.8.24
0x86	ResponseOnEvent	Internally	7.3.2.8.25

Table 7.2: UDS Services supported by Diagnostic Server instance

Note:

- UDS services which are not supported by DM, are documented in the section Known Limitations.
- Support Type Internally means, that the service with the given SID can be completely processed internally within the Diagnostic Server instance without relying on external functionality - typically in form of an AA. Support Type Externally means, that the Diagnostic Server instance needs to call

an external function, to be able to process the service with the given SID. The mixed support Type "Internally & Externally" means, that for the service with the given SID partially calls to an external function have to be done, but it partially could be also handled internally.

7.3.2.8.2 Common service processing items

This chapter contains rules for service processors, shared among multiple services.

Memory related UDS services (such as 0x34 RequestDownload) use the request parameter addressAndLengthFormatIdentifier to identify the number of bytes transmitted on the bus for memory address and size. Regardless of the wire representation of address and length information, within the Diagnostic Server instance and external service processors all addresses and data length information are mapped to a uint64 datatype.

[SWS_DM_00129] Supported addressAndLengthFormatIdentifier

Upstream requirements: RS_Diag_04120

⌈The Diagnostic Server instance shall support for each nibble of the addressAndLengthFormatIdentifier a value between 1 and 8.⌋

[SWS_DM_00130] Not supported addressAndLengthFormatIdentifier

Upstream requirements: RS_Diag_04120

⌈The Diagnostic Server instance shall send the negative response 0x31 (requestOutOfRange), if an addressAndLengthFormatIdentifier with a value outside the range between 1 and 8 is received.⌋

7.3.2.8.3 UDS service serialization

The DEXT model describes how UDS entities are transferred on the network. This includes the endianness of data types of DiagnosticDataElements. The DiagnosticDataElements allow big and little endian to be used for the transmission on the network. The used endianness type might or might not match with the endianness of the CPU. In case it does not match, the DM will automatically convert between the endianness that is required on the network (DEXT) and the data that is provided to/from DiagnosticDataElements.

[SWS_DM_01759] DiagnosticDataElement serialization respecting the data element endianness for typed interfaces

Upstream requirements: RS_Diag_04173

⌈The DM shall serialize or deserialize all DiagnosticDataElement that are read/written to a typed interface, respecting the endianness type that is provided by DiagnosticDataElement.swDataDefProps.baseType.baseTypeDefinition.byteOrder. If baseType.baseTypeDefinition.byteOrder is not configured for that DiagnosticDataElement, the DM shall use the default endianness provided by DiagnosticCommonProps.defaultEndianness.⌋

7.3.2.8.4 Service 0x10 – DiagnosticSessionControl

The UDS service DiagnosticSessionControl is used to enable different diagnostic sessions in the server.

[SWS_DM_00226] Support of UDS service DiagnosticSessionControl

Upstream requirements: RS_Diag_04198, RS_Diag_04248

⌈The Diagnostic Server instance shall provide the UDS service 0x10 DiagnosticSessionControl according to ISO 14229-1[1].⌋

[SWS_DM_00227] Check for supported sessions

Upstream requirements: RS_Diag_04196, RS_Diag_04248

⌈If the Subfunction addressed by the DiagnosticSessionControl according to [SWS_DM_00226] is not supported by the configuration, i.e., there is no DiagnosticSession configured with id matching the requested Subfunction value, the Diagnostic Server instance shall return a NRC 0x12 (SubFunctionNotSupported).⌋

In the context of parallel clients, a DiagnosticSessionControl may lead to negative responses even for supported Subfunctions with positive permission checks.

[SWS_DM_00228] Switch to requested Diagnostic Session

Upstream requirements: RS_Diag_04198, RS_Diag_04248

⌈On positive evaluation of a DiagnosticSessionControl request, the Diagnostic Server instance shall send the positive response message. After the response message is sent, the Diagnostic Server shall internally switch to the DiagnosticSession with id matching the requested Subfunction value, and shall set new timing parameters according to the associated parameters p2ServerMax and p2StarServerMax.⌋

[SWS_DM_00845] Notification about session change

Upstream requirements: RS_Diag_04208

⌈If the Diagnostic Server instance did successfully change the session of a conversation, it shall update the diagnostic session of the according ara::diag::Conversation class instance internally.⌋

7.3.2.8.5 Service 0x11 – ECUReset

The term ECUReset originates in the Classic Platform world and ISO 14229-1[1]. Regarding the Adaptive Platform, an ECUReset does not necessarily affect the whole ECU or machine, but must be interpreted in the context of the diagnostic address entity it targets, e. g. a platform or application level SoftwareCluster. Because the service name ECUReset is well known in diagnostics, it was decided to keep using the term throughout the affected Adaptive Platform specifications.

[SWS_DM_00234] Support of UDS service ECUReset

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall support the UDS service 0x11 ECU Reset according to ISO 14229-1[1].⌋

[SWS_DM_00269] Reaction on Unsupported Subfunction

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall send a negative response 0x12 (SubFunctionNotSupported), if the requested subfunction value is neither in configured range of default subfunction values (requestType, see ISO 14229-1[1]) nor in range of the configured DiagnosticEcuReset.customSubFunctionNumber in the ECU.⌋

7.3.2.8.6 RapidPowerShutdown sub – functions

[SWS_DM_01020] EnableRapidPowerShutdown processing

Upstream requirements: RS_Diag_04196

⌈If EnableRapidPowerShutdown request is received, the DM shall trigger a call of ara::diag::EcuResetRequest::EnableRapidShutdown with enable set to TRUE if it is configured as per [SWS_DM_00100].⌋

[SWS_DM_01092] EnableRapidPowerShutdown Positive Response

Upstream requirements: RS_Diag_04196

⌈If a positive response for subfunction 'EnableRapidPowerShutdown' shall be sent, the DM shall put the configured value of SoftwareClusterDiagnosticDeployment-Props.powerDownTime in its positive response.⌋

[SWS_DM_01021] DisableRapidPowerShutdown processing

Upstream requirements: RS_Diag_04196

⌈If DisableRapidPowerShutdown request is received, the DM shall trigger a call of ara::diag::EcuResetRequest::EnableRapidShutdown with enable set to FALSE if it is configured as per [SWS_DM_00100].⌋

7.3.2.8.7 All Other sub – functions

[SWS_DM_00235] ECUReset service processing

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall call the method ara::diag::EcuResetRequest::RequestReset of the ara::diag::EcuResetRequest class instance to request an ECU to the application Reset except for Enable-/DisableRapidPowerShutdown services.⌋

[SWS_DM_00268] EcuReset positive response processing before reset

Upstream requirements: RS_Diag_04019

⌈If the method ara::diag::EcuResetRequest::RequestReset did NOT raise an ApApplicationError, the Diagnostic Server instance shall trigger return a positive response before the actual reset, in case the parameter DiagnosticEcuResetClass.respondToReset is either not present or present and set to DiagnosticResponseToEcuResetEnum.respondBeforeReset.⌋

[SWS_DM_01023] Calling ExecuteReset() if positive response shall be sent before reset

Upstream requirements: RS_Diag_04196

⌈If the parameter DiagnosticEcuResetClass.respondToReset is either not present or present and set to respondBeforeReset and the Diagnostic Server instance has sent the positive response according to [SWS_DM_00268], the Diagnostic Server instance shall call ara::diag::EcuResetRequest::ExecuteReset.⌋

[SWS_DM_00360] EcuReset positive response processing after reset

Upstream requirements: RS_Diag_04196

⌈If the method `ara::diag::EcuResetRequest::RequestReset` did NOT raise an `ApApplicationError`, the `Diagnostic Server instance` shall trigger return a positive response after the actual reset (i.e. if after the `apext::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment` is called) in case the parameter `DiagnosticEcuResetClass.respondToReset` is present and set to `DiagnosticResponseToEcuResetEnum.respondAfterReset`.⌋

Note: The information, that the reset shall be transmitted after the `apext::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment` method is called can be stored by a flag in non-volatile memory.

[SWS_DM_01346] Handling negative return values of ara::diag::EcuResetRequest::ExecuteReset

Upstream requirements: RS_Diag_04196

⌈If `ara::diag::EcuResetRequest::ExecuteReset` returns any of the defined error codes in [SWS_DM_01014], the `Diagnostic Server instance` shall return a Negative Response with `NRC` equal to the returned error code, if a final response to the request was not sent yet.⌋

[SWS_DM_01347] Handling unspecified negative return values of ara::diag::EcuResetRequest::ExecuteReset

Upstream requirements: RS_Diag_04196

⌈If `ara::diag::EcuResetRequest::ExecuteReset` returns any error code other than the defined error codes in [SWS_DM_01014], the `Diagnostic Server instance` shall return a Negative Response with `NRC` equal to 0x10 (`generalReject`), if a final response to the request was not sent yet.⌋

Note: Additionally, the FunctionalCluster or Application processing the reset should trigger a Log and Trace message giving the detailed cause for the error code(e.g. sequence error: `ara::diag::EcuResetRequest::RequestReset` has not been called before `ara::diag::EcuResetRequest::ExecuteReset` for the corresponding instance).

[SWS_DM_01090] Calling ExecuteReset() if positive response shall be sent after reset

Upstream requirements: RS_Diag_04196

⌈If the parameter `DiagnosticEcuResetClass.respondToReset` is set to `respondAfterReset`, the Diagnostic Server instance shall directly call `ara::diag::`

EcuResetRequest::ExecuteReset without sending the positive response according to [SWS_DM_00268].⌋

Note: Sending the positive response after the reset is handled by [SWS_DM_00360].

[SWS_DM_01022] Block requests after ara::diag::EcuResetRequest::RequestReset called

Upstream requirements: RS_Diag_04196

⌈If the reset request is accepted , the Diagnostic Server instance that received the EcuReset request shall ignore further incoming UDS requests.⌋

[SWS_DM_01309] Unblock requests after ara::diag::EcuResetRequest::ExecuteReset completed

Upstream requirements: RS_Diag_04196

⌈If UDS requests are blocked due to [SWS_DM_01022] and the ara::core::Future returned from the call of ara::diag::EcuResetRequest::ExecuteReset, the DM shall accept incoming UDS requests again.⌋

7.3.2.8.8 Service 0x14 – ClearDiagnosticInformation

The UDS service ClearDiagnosticInformation is used to clear the ECUs fault memory.

[SWS_DM_00090] Support of UDS service ClearDiagnosticInformation

Upstream requirements: RS_Diag_04180, RS_Diag_04196

⌈The Diagnostic Server instance shall provide the UDS service 0x14 ClearDiagnosticInformation according to ISO 14229-1[1].⌋

[SWS_DM_00091] Evaluation of ClearDiagnosticInformation parameters

Upstream requirements: RS_Diag_04180, RS_Diag_04117

⌈The Diagnostic Server instance shall determine the DTC group or single DTC to clear from the 'groupOfDTC' parameter of the UDS request.⌋

[SWS_DM_00092] Parameter range check for groupOfDTC request parameter

Upstream requirements: RS_Diag_04180, RS_Diag_04117

⌈The Diagnostic Server instance shall reply with an NRC 0x31 (RequestOutOfRange) if the requested 'groupOfDTC' has no matching configured DTC group according to [SWS_DM_00064] or configured DTC by DiagnosticTroubleCodeUds.udsDtcValue.⌋

[SWS_DM_00113] Positive response for UDS service 0x14

Upstream requirements: RS_Diag_04196

⌈If the Diagnostic Server instance has cleared the requested 'groupOfDTC', the Diagnostic Server instance shall send a positive response.⌋

The DTC clearing behavior is described in detail in section 7.3.4.4.5. It consists of resetting the DTC status and deleting snapshot records and extended data records.

[SWS_DM_00115] Memory error handling while clearing DTCs

Upstream requirements: RS_Diag_04180

⌈The Diagnostic Server instance shall return a negative response NRC 0x72 (generalProgrammingFailure) if it encounters an error in the non-volatile memory while clearing the DTCs.⌋

The definition of a failure of the non-volatile memory is hardware and project specific. In general if the clear DTC operation could not delete the snapshot records, extended data records and if it could not reset the UDS DTC status byte because the underlying storage system reported and error, a non-volatile memory error can be assumed.

[SWS_DM_00122] UDS response behavior on not allowed clear operations

Upstream requirements: RS_Diag_04117

⌈If a DTC clear operation is requested and the DTC clear operation shall clear a DTC with a forbidden clear allowance according to [SWS_DM_00896], the Diagnostic Server instance shall send a negative response 0x22 (conditionsNotCorrect) in the following situations:

- it was requested to clear a single DTC and the DTC could not be cleared according to [SWS_DM_00896]
- it was requested to clear a DTC group and all the DTCs of the DTC group could not be cleared according to [SWS_DM_00896]
  (This doesn't apply when one or more DTC are allowed to be cleared.)

⌋

[SWS_DM_00159] Allow only to clear `GroupOfAllDTCs`

Upstream requirements: RS_Diag_04117

⌈If the configuration `DiagnosticMemoryDestination.clearDtcLimitation` is set to `clearAllDtcs`, the `Diagnostic Server instance` shall only allow to clear all DTCs via the `GroupOfAllDTC` as defined in [SWS_DM_00065]. In case a different value is given in groupOfDTC request parameter, the `Diagnostic Server instance` shall return a negative response 0x31 (`RequestOutOfRange`).⌋

[SWS_DM_00160] Allow to clear single `DTCs`

Upstream requirements: RS_Diag_04117

⌈If the configuration `DiagnosticMemoryDestination.clearDtcLimitation` is set to `allSupportedDtcs`, the `Diagnostic Server instance` shall allow to clear single DTCs or DTCGroups. [SWS_DM_00092] defines the possible and refused values.⌋

[SWS_DM_01578] Behavior of not configured `DiagnosticMemoryDestination.clearDtcLimitation`

Upstream requirements: RS_Diag_04117

⌈If the optional parameter `DiagnosticMemoryDestination.clearDtcLimitation` is not configured, the `DM` shall use a default value of `allSupportedDtcs` for that parameter.⌋

[SWS_DM_00162] Point in time for positive response for ClearDTC

Upstream requirements: RS_Diag_04180, RS_Diag_04196

⌈The `Diagnostic Server instance` shall send a positive response for a ClearDiagnosticInformation service after all memory is cleared in the server. This is regardless how the `Diagnostic Server instance` memory is organized (splitted, volatile, non-volatile).⌋

[SWS_DM_00163] Definition of a inhibited clear operation on single DTC

Upstream requirements: RS_Diag_04180

⌈If it is requested to clear a single DTC and a `DiagnosticClearCondition` referenced from this DTC (via `DiagnosticTroubleCodeUdsToClearConditionGroupMapping`) is not fulfilled, the `Diagnostic Server instance` shall send a negative response 0x22 (`conditionsNotCorrect`).⌋

[SWS_DM_00164] Definition of a inhibited clear operation for a group of DTCs

Upstream requirements: RS_Diag_04180

⌈If it is requested to clear a group of DTCs, the `Diagnostic Server instance` shall send a negative response 0x22 (`conditionsNotCorrect`) if all

DTCs of that group of DTC forbid the clearance according to [SWS_DM_00163] or [SWS_DM_00896].⌋

7.3.2.8.8.1 Clearing user-defined fault memory

According to [SWS_DM_00090] the Diagnostic Server instance implements an ISO 14229-1[1] compatible UDS service ClearDiagnosticInformation.

The upcoming subchapter refers to ISO 14229-1:2020 [1].

The clearance of a user-defined fault memory has the same behavior as the clearing of the primary fault memory. All requirements that are provided to clear the primary fault memory also apply to a clear of a user-defined fault memory.

[SWS_DM_00193] Support of a user-defined fault memory clear request

Upstream requirements: RS_Diag_04197

⌈If the Diagnostic Server instance receives a a UDS service 0x14 ClearDiagnosticInformation with a length of 5 bytes, the Diagnostic Server instance shall interpret this request as a request to clear user-defined fault memory.⌋

[SWS_DM_00194] Definition of the user-defined fault memory number for ClearDiagnosticInformation

Upstream requirements: RS_Diag_04197

⌈If the Diagnostic Server instance receives a UDS request to clear user-defined fault memory according to [SWS_DM_00193], the DM shall get the number of user-defined fault memory to be cleared from the fifth byte in the request.⌋

[SWS_DM_00195] Clearing a user-defined memory

Upstream requirements: RS_Diag_04197

⌈If the Diagnostic Server instance is requested to clear the user-defined fault memory according to [SWS_DM_00193] and an DiagnosticMemoryDestinationUserDefined.memoryId exists with the requested user-defined memory number according to [SWS_DM_00194], the Diagnostic Server instance shall clear the requested user-defined fault memory.⌋

For details about the fault memory clearing process please also refer to section 7.3.4.4.5.

[SWS_DM_00208] Validation of the requested user-defined memory number

Upstream requirements: RS_Diag_04197

⌈If the Diagnostic Server instance is requested to clear the user-defined fault memory according to [SWS_DM_00193] and no DiagnosticMemoryDestinationUserDefined.memoryId exists with the requested user-defined memory number according to [SWS_DM_00194], the Diagnostic Server instance shall return a NRC 0x31 (RequestOutOfRange).⌋

7.3.2.8.9 Service 0x19 – ReadDTCInformation

Some UDS responses for the Service "0x19 – ReadDTCInformation" use the parameter "DTCFormatIdentifier" as part of the response PDU. The Diagnostic Server instance obtains the value used from the global configuration item DiagnosticMemoryDestinationPrimary.typeOfDtcSupported. To provide the correct UDS values, the following mapping is used:

[SWS_DM_00062] Mapping between ISO 14229-1 and Autosar Diagnostic Extract Template of the `DTCFormatIdentifier`

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈If a positive response for service 0x19 with the ISO 14229-1[1] parameter "DTCFormatIdentifier" is sent, the Diagnostic Server instance shall derive the value from DiagnosticMemoryDestinationPrimary.typeOfDtcSupported applying the following mapping rule:⌋

typeOfDtcSupported	"DTCFormatIdentifier"
iso11992_4	0x03
iso14229_1	0x01
saeJ2012_da	0x00

[SWS_DM_00966] Reporting of DTCStatusAvailabilityMask

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈For all positive response for service 0x19 with DTCStatusAvailabilityMask in the response, the DM shall use the configured value from dtcStatusAvailabilityMask.⌋

7.3.2.8.9.1 SF 0x01 – reportNumberOfDTCByStatusMask

[SWS_DM_00244] Support of UDS service ReadDTCInformation, Subfunction 0x01

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x01 (reportNumberOfDTCByStatusMask) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_NUMBER_OF_DTC_BY_STATUS_MASK'.⌋

[SWS_DM_00061] Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask

Upstream requirements: RS_Diag_04157, RS_Diag_04067

⌈While sending the positive response for ReadDTCInformation.reportNumberOfDTCByStatusMask, the Diagnostic Server instance shall set the response PDU "DTCFormatIdentifier" according to the mapping of [SWS_DM_00062].⌋

7.3.2.8.9.2 SF 0x02 – reportDTCByStatusMask

[SWS_DM_00245] Support of UDS service ReadDTCInformation, Subfunction 0x02

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x02 (reportDTCByStatusMask) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_DTC_BY_STATUS_MASK'.⌋

7.3.2.8.9.3 SF 0x03 – reportDTCSnapshotIdentification

[SWS_DM_01256] Support of UDS service ReadDTCInformation, Subfunction 0x03

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x03 (reportDTCSnapshotIdentification) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_DTC_SNAPSHOT_IDENTIFICATION'.⌋

7.3.2.8.9.4 SF 0x04 – reportDTCSnapshotRecordByDTCNumber

[SWS_DM_00246] Support of UDS service ReadDTCInformation, Subfunction 0x04

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067, RS_Diag_04244

⌈The Diagnostic Server instance shall support Subfunction 0x04 (reportDTCSnapshotRecordByDTCNumber) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_DTC_SNAPSHOT_RECORD_BY_DTC_NUMBER'.⌋

7.3.2.8.9.5 SF 0x06 – reportDTCExtDataRecordByDTCNumber

[SWS_DM_00370] Support of UDS service ReadDTCInformation, Subfunction 0x06

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067, RS_Diag_04245

⌈The Diagnostic Server instance shall support Subfunction 0x06 (reportDTCExtDataRecordByDTCNumber) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_DTC_EXT_DATA_RECORD_BY_DTC_NUMBER'.⌋

7.3.2.8.9.6 SF 0x07 – reportNumberOfDTCBySeverityMaskRecord

[SWS_DM_00247] Support of UDS service ReadDTCInformation, Subfunction 0x07

Upstream requirements: RS_Diag_04180, RS_Diag_04157

⌈The Diagnostic Server instance shall support Subfunction 0x07 (reportNumberOfDTCBySeverityMaskRecord) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_NUMBER_OF_DTC_BY_SEVERITY_MASK_RECORD'.⌋

[SWS_DM_00063] Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord

Upstream requirements: RS_Diag_04157, RS_Diag_04067

⌈While sending the positive response for ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord, the Diagnostic Server instance shall set the response PDU "DTCFormatIdentifier" according to the mapping of [SWS_DM_00062].⌋

7.3.2.8.9.7 SF 0x0A – reportSupportedDTC

[SWS_DM_00967] Support of UDS service ReadDTCInformation, Subfunction 0x0A

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈If a DiagnosticReadDTCInformation of category 'REPORT_SUPPORTED_DTC exists, the Diagnostic Server instance shall support subfunction 0x0A (reportSupportedDTC) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1].⌋

[SWS_DM_00968] Reporting of DTCAndStatusRecord parameter

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The *DTCAndStatusRecord* parameter according to ISO 14229-1[1] as part of the response shall consist of pairs of DTC number and its according DTC status of all supported DTCs of the DMs primary memory with no fixed and specified order.⌋

7.3.2.8.9.8 SF 0x14 – reportDTCFaultDetectionCounter

[SWS_DM_00371] Support of UDS service ReadDTCInformation, Subfunction 0x14

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x14 (reportDTCFaultDetectionCounter) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_DTC_FAULT_DETECTION_COUNTER'.⌋

7.3.2.8.9.9 SF 0x17 – reportUserDefMemoryDTCByStatusMask

[SWS_DM_00372] Support of UDS service ReadDTCInformation, Subfunction 0x17

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x17 (reportUserDefMemoryDTCByStatusMask) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_USER_DEF_MEMORY_DTC_BY_STATUS_MASK'.⌋

7.3.2.8.9.10 SF 0x18 – reportUserDefMemoryDTCSnapshotRecordByDTCNumber

[SWS_DM_00373] Support of UDS service ReadDTCInformation, Subfunction 0x18

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x18 (reportUserDefMemoryDTCSnapshotRecordByDTCNumber) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_USER_DEF_MEMORY_DTC_SNAPSHOT_RECORD_BY_DTC_NUMBER'.⌋

7.3.2.8.9.11 SF 0x19 – reportUserDefMemoryDTCExtDataRecordByDTCNumber

[SWS_DM_00374] Support of UDS service ReadDTCInformation, Subfunction 0x19

Upstream requirements: RS_Diag_04180, RS_Diag_04157, RS_Diag_04067

⌈The Diagnostic Server instance shall support Subfunction 0x19 (reportUserDefMemoryDTCExtDataRecordByDTCNumber) of the UDS service 0x19 ReadDTCInformation according to ISO 14229-1[1], provided the configuration contains a DiagnosticReadDTCInformation of category 'REPORT_USER_DEF_MEMORY_DTC_EXT_DATA_RECORD_BY_DTC_NUMBER'.⌋

7.3.2.8.10 Service 0x22 – ReadDataByIdentifier

The processing of a UDS Service ReadDataByIdentifier (0x22) is described in ISO 14229-1[1], see in particular the evaluation sequence in Figure 15. On processing, the Diagnostic Server instance needs to perform various checks. The following requirements determine the relation between the input data to be checked and the configuration provided to the Diagnostic Server instance via DEXT parameters.

[SWS_DM_00170] Realisation of UDS service ReadDataByIdentifier (0x22)

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the diagnostic service 0x22 ReadDataByIdentifier according to ISO 14229-1[1].⌋

[SWS_DM_00412] Check requested number of DataIdentifiers

Upstream requirements: RS_Diag_04203, RS_Diag_04228

⌈On reception of the UDS Service ReadDataByIdentifier (0x22), the Diagnostic Server instance shall check the number of the requested DataIdentifiers within one message. In case a UDS Service ReadDataByIdentifier (0x22) request contains more DataIdentifiers than defined by maxDidToRead, the request shall be rejected with NRC (0x13).⌋

[SWS_DM_00409] Check supported DataIdentifier

Upstream requirements: RS_Diag_04203

⌈On reception of the UDS Service ReadDataByIdentifier (0x22), a requested DataIdentifier shall be considered as supported if and only if there exists a DiagnosticDataIdentifier with id matching the DataIdentifier and this DiagnosticDataIdentifier is referenced by a DiagnosticReadDataByIdentifier.⌋

[SWS_DM_00413] Check supported DataIdentifier in active session

Upstream requirements: RS_Diag_04203, RS_Diag_04226

⌈On reception of the UDS Service ReadDataByIdentifier (0x22), a requested DataIdentifier shall be considered as supported in active session if and only if the DataIdentifier is supported according to [SWS_DM_00409] and the DiagnosticDataByIdentifier.accessPermission references by its DiagnosticAccessPermission.diagnosticSession the active diagnostic session in the DM.⌋

[SWS_DM_00414] Check supported DataIdentifier on active security level

Upstream requirements: RS_Diag_04203, RS_Diag_04227

⌈On reception of the UDS Service ReadDataByIdentifier (0x22), a requested DataIdentifier shall be considered as supported on active security level if and only if

the DataIdentifier is supported according to [SWS_DM_00409] and the DiagnosticDataByIdentifier.accessPermission references by its DiagnosticAccessPermission.securityLevel the active security level in the DM.⌋

[SWS_DM_00570] Retrieving data for requested DataIdentifier

Upstream requirements: RS_Diag_04097

⌈On reception of the UDS Service ReadDataByIdentifier (0x22), the Diagnostic Server instance shall retrieve the mapped data for that DataIdentifier.

There are various ways how DID data retrieving is modeled. Among them are:

- received entire DID via ara::diag interface [SWS_DM_00601] or [SWS_DM_00607]
- received data for data elements via ara::diag interface [SWS_DM_00603]
- generic UDS service processing [SWS_DM_00602]
- internal data [SWS_DM_00393]

⌋

Note: Also, a DiagnosticDataIdentifier's single dataElement (referenced by a meta-class DiagnosticDataElement) can be accessed from the associated RPortPrototype. Refer to chapters 7.3.6.2.1 and 7.3.6.2.2 for more details.

Note: The presence of an ara::diag::DiagUdsNrcErrorDomain in the Result already indicates a negative result of the external diagnostic processor. Especially for multi DID requests a single failure might still lead to a positive response if at least one DID is supported in the active session.

7.3.2.8.11 Service 0x27 – SecurityAccess

[SWS_DM_00236] Realization of UDS service 0x27 SecurityAccess

Upstream requirements: RS_Diag_04196, RS_Diag_04005

⌈The Diagnostic Server instance shall implement the diagnostic service 0x27 SecurityAccess according to ISO 14229-1[1].⌋

[SWS_DM_00863] Checking Supported Subfunction for RequestSeed

Upstream requirements: RS_Diag_04203

⌈On reception of a request for UDS Service SecurityAccess (0x27), the Diagnostic Server instance shall call ara::diag::SecurityAccess::GetSeed if the requested subfunction value (securityAccessType) matches to the value of the instance of DiagnosticSecurityAccess with requestSeedId. The securityAccessDataRecord parameter shall be filled with the securityAccessDataRecord

provided by the Diagnostic Client. If no data is provided by the Diagnostic Client, the securityAccessDataRecord parameter shall be empty.⌋

Note: The static seed mechanism, as specified in ISO 14229-1[1] - annex I.2 table I.1, needs to be done by the application with the implementation of ara::diag::SecurityAccess::GetSeed and ara::diag::SecurityAccess::CompareKey.

[SWS_DM_00507] Length check on UDS Service 0x27 request with Subfunction for RequestSeed

Upstream requirements: RS_Diag_04203, RS_Diag_04228

⌈On reception of a request for UDS Service SecurityAccess (0x27) with subfunction value matching the requestSeedId of a configured DiagnosticSecurityAccess, the Diagnostic Server instance shall perform the message length check against the optionally configured accessDataRecordSize of the related DiagnosticSecurityLevel. A non-present parameter accessDataRecordSize results in a check against 0 additional request bytes. If the length check fails, the Diagnostic Server instance shall send NRC 0x13 (IncorrectMessageLengthOrInvalidFormat).⌋

[SWS_DM_00864] Checking Supported Subfunction for CompareKey

Upstream requirements: RS_Diag_04203

⌈The Diagnostic Server instance shall call ara::diag::SecurityAccess::CompareKey when the requested subfunction value (securityAccessType) - 1 (to get the corresponding requestSeed) is equal to the value of instance of DiagnosticSecurityAccess with requestSeedId.⌋

[SWS_DM_00363] Unsupported Subfunction

Upstream requirements: RS_Diag_04196

⌈If the requested subfunction value is not configured (no instances of DiagnosticSecurityAccess with requestSeedId, as well as the corresponding CompareKey values), a negative response 0x12 (SubFunctionNotSupported) shall be returned.⌋

[SWS_DM_00846] Notification about security-level change

Upstream requirements: RS_Diag_04208

⌈If the Diagnostic Server instance did successfully change the security-level of a conversation, it shall update the security level of according ara::diag::Conversation class instance internally. Whether a security level is applicable by the DiagnosticSecurityAccess is defined by securityLevel.⌋

[SWS_DM_00270] Counting of attempts to change security level

Upstream requirements: RS_Diag_04005

⌈The Diagnostic Server instance shall count the number of failed attempts to change a requested security level. The Counter shall be reset if the security level change has passed successfully.⌋

[SWS_DM_00271] Evaluate the number of failed security level change attempts

Upstream requirements: RS_Diag_04005

⌈The Diagnostic Server instance shall compare the number of failed DiagnosticSecurityLevel changes with threshold value numFailedSecurityAccess after each failed attempt.
If the number of failed attempts is below the threshold value numFailedSecurityAccess the Diagnostic Server instance shall send a negative response with NRC 0x35 (InvalidKey).
If the number of failed attempts reaches the threshold value numFailedSecurityAccess the Diagnostic Server instance shall start a delay timer configured with value securityDelayTime (see [SWS_DM_00272]) and send a negative response with NRC 0x36 (exceededNumberOfAttempts).
In both cases a DiagnosticSecurityLevel change must not be done if the attempt failed before.⌋

The delay timer represents the required minimum time between security access attempts, after one time negative response with NRC 0x36 (exceededNumberOfAttempts) was sent out.

[SWS_DM_00272] Expiration of the delay timer

Upstream requirements: RS_Diag_04005

⌈As long as the delay timer (see [SWS_DM_00271]) configured with threshold value securityDelayTime has not expired, all requests for DiagnosticSecurityLevel change with subfunction value (access type) requestSeed shall be responded with NRC 0x37 (requiredTimeDelayNotExpired).
⌋

[SWS_DM_00478] Persistent Storage of failed attempts to change security level

Upstream requirements: RS_Diag_04005

⌈The Diagnostic Server instance shall store the number of failed attempts persistently for every security access type separately. (see [SWS_DM_00270])⌋

[SWS_DM_00479] Security Access Delay Timer on power up when attempt counter threshold is reached

Upstream requirements: RS_Diag_04005

⌈If at least one of the restored failed attempt counters is greater or equal to the threshold value DiagnosticSecurityLevel.numFailedSecurityAccess the Diagnostic Server instance shall restart the security delay timer with the higher value of DiagnosticSecurityAccess.securityDelayTimeOnBoot/ DiagnosticSecurityLevel.securityDelayTime of the according DiagnosticSecurityLevel .⌋

[SWS_DM_01758] Security Access Delay Timer on power up

Upstream requirements: RS_Diag_04005

⌈If the restored failed attempt counter is not greater or equal to the threshold value DiagnosticSecurityLevel. numFailedSecurityAccess,the Diagnostic Server instance shall restart the security delay timer with the value of DiagnosticSecurityAccess.securityDelayTimeOnBoot of the according DiagnosticSecurityLevel.⌋

[SWS_DM_00480] Shared Security Access Delay Timer

Upstream requirements: RS_Diag_04005

⌈If DiagnosticSecurityAccessClass.sharedTimer is set to true, a shared delay timer instance shall be used for all security levels.⌋

The effect of a started security access delay timer is always according to [SWS_DM_00272].

7.3.2.8.12 Service 0x28 – CommunicationControl

[SWS_DM_00140] Realisation of UDS service 0x28 CommunicationControl

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the diagnostic service 0x28 CommunicationControl according to ISO 14229-1[1].⌋

[SWS_DM_00252] Reaction on Unsupported Subfunction

Upstream requirements: RS_Diag_04203

⌈The Diagnostic Server instance shall check, whether the Subfunction addressed by the CommunicationControl is supported by an existing DiagnosticComControl.category in the configuration and allow further processing. If the Subfunction addressed by the CommunicationControl is not supported by an existing DiagnosticComControl.category in the configuration a negative response 0x12 (SubFunctionNotSupported) shall be returned.⌋

[SWS_DM_00865] Communication control service processing

Upstream requirements: RS_Diag_04169

⌈The Diagnostic Server instance shall call the method ara::diag::CommunicationControl::CommCtrlRequest to process a communication control service.⌋

[SWS_DM_00866] Negative Response processing

Upstream requirements: RS_Diag_04196

⌈If the external processor raised an error according to ara::diag::DiagUdsNrcErrc, the Diagnostic Server instance shall return a negative response with the value of that error code.⌋

[SWS_DM_00199] Positive Response processing

Upstream requirements: RS_Diag_04196

⌈If the external processor did raise no ApApplicationError, the Diagnostic Server instance shall return a positive response.⌋

7.3.2.8.13 Service 0x29 – Authentication

The UDS service Authentication (0x29) is used by the ECU as a means to identify the client and provide the relevant access to diagnostic resources, based on the client's 'role'. The access to these diagnostic resources can be limited in time, or, bound to certain vehicles or ECUs. The specifications of this chapter are based on the UDS Specifications ISO 14229-1:2020 [1]. The specifications of this chapter are a sub-set of the service 0x29 specifications of ISO 14229-1:2020 [1], and the Diagnostic Manager may only implement the specifications of this chapter.

[SWS_DM_01226] Support of UDS service authentication

Upstream requirements: RS_Diag_04251

⌈If configured, the Diagnostic Server instance shall provide the UDS service 0x29 Authentication with the subfunctions.

- deAuthenticate
- verifyCertificateUnidirectional
- verifyCertificateBidirectional
- proofOfOwnership
- transmitCertificate
- authenticationConfiguration

according to ISO 14229-1:2020 [1].⌋

Note: The Diagnostic Manager only implements the authentication via PKI certificate exchange. Authentication with challenge-response (ACR) is currently out of scope of the Diagnostic Manager.

[SWS_DM_01227] Configuration of authentication types

Upstream requirements: RS_Diag_04251

⌈If the sub function DiagnosticProofOfOwnership of the `DiagnosticAuthentication` `DiagnosticServiceInstance` is configured, the Diagnostic Manager shall mandatorily require one of the following sub functions to be configured as well:

- DiagnosticVerifyCertificateUnidirectional
- DiagnosticVerifyCertificateBidirectional

⌋

[SWS_DM_01228] Mandatory sub functions

Upstream requirements: RS_Diag_04251

⌈If one of the following sub functions of `DiagnosticAuthentication`, `DiagnosticServiceInstance` is configured:

- DiagnosticVerifyCertificateUnidirectional
- DiagnosticVerifyCertificateBidirectional
- DiagnosticProofOfOwnership

the Diagnostic Manager shall mandatorily require the following sub functions to be configured as well:

- DiagnosticDeAuthenticate
- DiagnosticAuthenticationConfiguration

⌋

7.3.2.8.13.1 VerifyCertificateUnidirectional

[SWS_DM_01230] Processing the verifyCertificateUnidirectional request

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to verifyCertificateUnidirectional, if all checks described in [SWS_DM_00096] are successfully completed, the `Diagnostic Server instance` shall call the method `ara::diag::Authentication::VerifyCertificateUnidirectional` of the class `ara::diag::Authentication`. The `Diagnostic Server instance` shall pass the received communicationConfiguration (COCO), certificateClient (CECL) and challengeClient (CHCL) to the application in the parameters CommunicationConfiguration, ClientCertificate and ClientChallenge of the method respectively.⌋

[SWS_DM_01231] Handling Negative return values of `ara::diag::Authentication::VerifyCertificateUnidirectional` method

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01230] returns any of the defined error codes in [SWS_DM_01126], the `Diagnostic Server instance` shall return a Negative Response with `NRC` equal to the returned error code.⌋

[SWS_DM_01233] Successful verification of verifyCertificateUnidirectional

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01230] executes successfully and returns no error code, the `Diagnostic Server instance` shall return a Positive Response to the tester. The field lengthOfChallengeServer (LOCHSE) and lengthOfEphemeralPublicKeyServer (LOEPKSE) shall be derived by the `Diagnostic Server instance` from the returned Challenge and filled in the Positive Response. The `Diagnostic Server instance` shall copy the returned Challenge into the response field challengeServer (CHSE) and the returned Ephemeral Public Key into the response field ephemeralPublicKeyServer (EPKSE). The `Diagnostic Server instance` shall set the parameter authenticationReturnParameter (RV) to 0x11 (CertificateVerified, OwnershipVerificationNecessary) in the positive Response.⌋

NOTE: The `Diagnostic Server instance` shall treat each verifyCertificateUnidirectional sub function request individually and shall not keep track of previously received verifyCertificateUnidirectional sub function requests. For E.g., when multiple verifyCertificateUnidirectional requests are received without the proofOfOwnership sub-function request.

7.3.2.8.13.2 VerifyCertificateBidirectional

[SWS_DM_01235] Processing the verifyCertificateBidirectional request

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to verifyCertificateBidirectional, if all checks described in [SWS_DM_00096] are successfully completed, the `Diagnostic Server instance` shall call the method `ara::diag::Authentication::VerifyCertificateBidirectional` of the class `ara::diag::Authentication`. The `Diagnostic Server instance` shall pass the received communicationConfiguration(COCO), certificateClient (CECL) and challengeClient (CHCL) to the application in the parameters CommunicationConfiguration, ClientCertificate and ClientChallenge of the method respectively.⌋

[SWS_DM_01236] Handling Negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01235] returns any of the defined error codes in [SWS_DM_01127], the `Diagnostic Server instance` shall return a Negative Response with `NRC` equal to the returned error code.⌋

[SWS_DM_01238] Successful verification of verifyCertificateBidirectional

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01235] executes successfully and returns no error code, the Diagnostic Manager shall return a Positive Response to the tester. The fields lengthOfChallengeServer (LOCHSE), lengthOfCertificateServer (LOCESE) and lengthOfProofOfOwnershipServer (LPOWNSE) and lengthOfEphemeralPublicKeyServer(LOEPKSE) shall be derived by the `Diagnostic Server instance` from the returned Challenge, Certificate, ProofOfOwnership and EphemeralPublicKey and filled in the Positive Response. The `Diagnostic Server instance` shall copy the returned Challenge into the response field challengeServer (CHSE), the returned Certificate into the response field certificateServer (CESE), the returned ProofOfOwnership into the response field proofOfOwnershipServer (POWNSE) and the returned Ephemeral Public Key into the response field ephemeralPublicKeyServer (EPKSE). The `Diagnostic Server instance` shall set the parameter authenticationReturnParameter (RV) to 0x11 (CertificateVerified, OwnershipVerificationNecessary) in the positive Response.⌋

NOTE: The `Diagnostic Server instance` shall treat each verifyCertificateBidirectional sub function request individually and shall not keep track of previously received verifyCertificateBidirectional sub function requests. For E.g, when multiple verifyCertificateBidirectional requests are received without the proofOfOwnership subfunction request

7.3.2.8.13.3 ProofOfOwnership

[SWS_DM_01240] Processing the proofOfOwnership request

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to proofOfOwnership, if all checks described in [SWS_DM_00096] are successfully completed, the `Diagnostic Server instance` shall call the method `ara::diag::Authentication::VerifyOwnership` of the class `ara::diag::Authentication`. The `Diagnostic Server instance` shall pass the received proofOfOwnershipClient (POWNCL) and ephemeralPublicKeyClient (EPKCL) to the application in the parameters ClientPOWN and ClientEphemeralPublicKey of the method respectively.⌋

[SWS_DM_01241] Handling Negative return values of `ara::diag::Authentication::VerifyOwnership` method

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01240] returns any of the defined error codes in [SWS_DM_01128], the `Diagnostic Server instance` shall return a Negative Response with `NRC` equal to the returned error code.⌋

[SWS_DM_01243] Successful verification of Client proofOfOwnership

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01240] executes successfully and returns no error code, the `Diagnostic Server instance` shall return a Positive Response to the tester. The fields lengthOfSessionKeyInfo (LOSKI), shall be derived by the `Diagnostic Server instance` from the returned SessionKeyInfo and filled in the Positive Response The `Diagnostic Server instance` shall copy the returned SessionKeyInfo into the response field sessionKeyInfo (SKI). The `Diagnostic Server instance` shall set the parameter authenticationReturnParameter (RV) to 0x12 (OwnershipVerified,AuthenticationComplete) in the positive Response.⌋

Once a positive or negative response has been sent to the tester for a proofOfOwnership sub-function request, the authentication sequence shall be deemed to be complete, by the `Diagnostic Server instance`. A new authentication sequence must be started with a verifyCertificateUnidirectional or verifyCertificateBidirectional sub function request.

7.3.2.8.13.4 DeAuthenticate

[SWS_DM_01244] Processing the deAuthenticate request

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to deAuthenticate, if all checks described in [SWS_DM_00096] are successfully completed, the Diagnostic Server instance shall perform the steps described in [SWS_DM_01212].⌋

[SWS_DM_01245] Successful completion of deAuthenticate

Upstream requirements: RS_Diag_04251

⌈If the requirement in [SWS_DM_01244] is executed successfully, the Diagnostic Server instance shall return a Positive Response to the tester. The Diagnostic Server instance shall set the parameter authenticationReturnParameter (RV) to 0x10 (Deauthentication Successful) in the positive Response.⌋

7.3.2.8.13.5 AuthenticationConfiguration

[SWS_DM_01246] Processing the authenticationConfiguration request

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to authenticationConfiguration, if all checks described in [SWS_DM_00096] are successfully completed, the Diagnostic Server instance shall return a positive response. The Diagnostic Server instance shall set the parameter authenticationReturnParameter (RV) to 0x02 (AuthenticationConfiguration APCE) in the positive Response.⌋

7.3.2.8.13.6 TransmitCertificate

[SWS_DM_01247] Validation of the transmitCertificate certificateEvaluationId

Upstream requirements: RS_Diag_04251

⌈On reception of an Authentication (0x29) service with sub-function equal to transmitCertificate, if all checks described in [SWS_DM_00096] are successfully completed, the Diagnostic Server instance shall validate if the received request parameter certificateEvaluationId (CEID) is supported by the DiagnosticAuthTransmitCertificate.DiagnosticAuthCertificateEvaluation.evaluationId configuration. If the parameter is not supported, the Diagnostic Server instance shall send a Negative Response with NRC 0x31 (Request outOfRange).⌋

The certificateEvaluationId of subfunction TransmitCertificate allows to distinguish between different types of certificates with own use cases. The Diagnostic Server instance uses a generic interface approach, where one instance of the ara::diag::TransmitCertificate class can handle one ore more certificates. It is possible that there is one ara::diag::TransmitCertificate interface by certificateEvaluationId, or any number of certificateEvaluationId handled by the interface. This gives the maximum of freedom to application developers.

[SWS_DM_01248] Processing the transmitCertificate request

Upstream requirements: RS_Diag_04251

⌈If the checks described in [SWS_DM_01247] are successfully completed, the Diagnostic Server instance shall call the method ara::diag::TransmitCertificate::Process. The Diagnostic Server instance shall pass the received certificateEvaluationId (CEID) and certificateData (CEDA) in the parameters certificateEvaluationId and certificateData of the method.⌋

[SWS_DM_01251] Successful verification of transmitCertificate

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01248] executes successfully and returns no error code, the Diagnostic Server instance shall return a Positive Response to the tester. The Diagnostic Server instance shall set the parameter authenticationReturnParameter (RV) to 0x13 (CertificateVerified) in the positive Response.⌋

[SWS_DM_01249] Handling Negative return values of ara::diag::TransmitCertificate::Process method

Upstream requirements: RS_Diag_04251

⌈If the method called in [SWS_DM_01248] returns any of the defined error codes in [SWS_DM_01968], the Diagnostic Server instance shall return a Negative Response with NRC equal to the returned error code.⌋

7.3.2.8.14 Service 0x2A – ReadDataByPeriodicIdentifier

The processing of a UDS Service ReadDataByPeriodicIdentifier (0x2A) is described in ISO 14229-1[1]. On processing, the Diagnostic Server instance needs to perform various checks. The following requirements determine the relation between the input data to be checked and the configuration provided to the Diagnostic Server instance via DEXT parameters.

[SWS_DM_01040] Realization of UDS service ReadDataByPeriodicIdentifier(0x2A)

Upstream requirements: RS_Diag_04215

⌈The Diagnostic Server instance shall implement the diagnostic service 0x2A ReadDataByPeriodicIdentifier according to ISO 14229-1[1].⌋

[SWS_DM_01041] Check requested number of periodic DataIdentifiers

Upstream requirements: RS_Diag_04215, RS_Diag_04228

⌈If the DM receives the UDS Service ReadDataByPeriodicIdentifier (0x2A) and the number of the requested PeriodicDataIdentifiers is larger than DiagnosticReadDataByPeriodicIDClass.maxPeriodicDidToRead, the DM shall return NRC 0x13 (incorrectMessageLengthOrInvalidFormat).⌋

[SWS_DM_01042] Minimum length check for ReadDataByPeriodicIdentifier

Upstream requirements: RS_Diag_04215, RS_Diag_04228

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), the DM shall check the request minimum length. If length of the request is less than 3 bytes for subfunctions different to 'stopSending' or less than 2 bytes, the DM shall respond with NRC 0x13 (incorrectMessageLengthOrInvalidFormat).⌋

[SWS_DM_01043] Check supported periodic DataIdentifier

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), the DM shall consider a requested PeriodicDataIdentifier as supported if and only if there exists a DiagnosticDataIdentifier in the range between 0xF200 and 0xF2FF with id matching the PeriodicDataIdentifier. If none of the requested Periodic DIDs are supported, the DM shall respond with NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01044] Check Transmission Mode

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) and if the requested transmission mode is different to a configured DiagnosticPeriodicRate.periodicRateCategory or to 'stopSending' , the DM shall respond with NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01045] Check Scheduler Availability

Upstream requirements: RS_Diag_04215

⌈If the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode different than 0x04 'stopSending' is received, and the number existing PDIDs and the new PDIDs from the request is larger than DiagnosticReadDataByPeriodicIDClass.schedulerMaxNumber , the DM shall respond with NRC 0x31(requestOutOfRange).⌋

[SWS_DM_01046] Check supported DataIdentifier in active session

Upstream requirements: RS_Diag_04215, RS_Diag_04226

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than 'stopSending', a requested Periodic DataIdentifier shall be considered as supported if and only if the active session passes the execution permission check according to [SWS_DM_00413] else process next periodic DID. If Session not supported for none of the periodic DIDs DM shall respond with NRC 0x31(requestOutOfRange).⌋

[SWS_DM_01047] Check supported DataIdentifier on active security level

Upstream requirements: RS_Diag_04215, RS_Diag_04227

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than stopSending, a requested PeriodicDataIdentifier shall be considered as supported on active security level if and only if the DataIdentifier the active security level passes the execution permission check according to [SWS_DM_00414] else the DM shall respond with NRC 0x33 (securityAccessDenied).⌋

[SWS_DM_01048] Check DataIdentifier for environmental conditions

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than stopSending, a requested PeriodicDataIdentifier shall be considered as supported if and only if the DataIdentifiers environmentalCondition allow an execution according to [SWS_DM_00112] else the DM shall respond with the NRC according to [SWS_DM_00289].⌋

[SWS_DM_01049] Checks Dynamically Defined DIDs in ReadDataByPeriodicIdentifier

Upstream requirements: RS_Diag_04215

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to true and the UDS Service ReadDataByPeriodicIdentifier (0x2A) is received with transmission- Mode different than stopSending, if verification has been successfully done according to [SWS_DM_01046], [SWS_DM_01047] and [SWS_DM_01048] and if the request contains one or more dynamically defined DID(s), the DM shall do the session, security checks and environmental condition checks for all source data.⌋

[SWS_DM_01050] Periodic DID length check

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than stopSending, the DM shall check if the length of each

requested PeriodicDataIdentifier is smaller or equal than the value provided by the periodic transmission handler apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength and if the size is exceeded for one or more PeriodicDataIdentifier, the DM shall return NRC 0x14 (responseTooLong).⌋

[SWS_DM_01051] DM behavior on transmission Mode stopSending without periodicDataIdentifier in the request

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode set to 'stopSending' and no periodicDataIdentifier in the request, the DM shall stop all scheduled periodicDataIdentifier transmissions.⌋

[SWS_DM_01052] DM behavior on transmission Mode stopSending with supported periodicDataIdentifier in the request

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode set to 'stopSending' and more than one supported periodicDataIdentifier is in the request, the DM shall stop the supported scheduled periodic data transmissions for all requested periodicDataIdentifiers.⌋

[SWS_DM_01053] DM behavior on transmission Mode stopSending with not supported periodicDataIdentifier in the request

Upstream requirements: RS_Diag_04215

⌈On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode set to 'stopSending' and none of the periodicDataIdentifiers of the request is supported, the DM shall return NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01054] Starting to transmit PDIDs after positive response

Upstream requirements: RS_Diag_04215

⌈After the positive response of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode different than stopSending was sent, the DM shall start to send the periodic DIDs.⌋

[SWS_DM_01055] Reaction on ApplicationError

Upstream requirements: RS_Diag_04215

⌈If an external processor to read the DID data raises an error of ara::diag::DiagUdsNrcErrorDomain, the DM shall skip the scheduled periodic response for the periodic DataIdentifier.⌋

[SWS_DM_01056] Optional condition checks for sending periodic DIDs

Upstream requirements: RS_Diag_04215, RS_Diag_04226, RS_Diag_04227

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to TRUE, the DM shall transmit the PDID only if the session, security and mode checks were executed successfully.⌋

[SWS_DM_01057] Optional stopping PDIDs after session change

Upstream requirements: RS_Diag_04215

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to TRUE and the DM is changing the diagnostic session, the DM shall remove all PDIDs from the list of scheduled PDIDs that are not allowed to be read in the new session.⌋

[SWS_DM_01058] Optional stopping PDIDs after security level change

Upstream requirements: RS_Diag_04215

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to TRUE and the DM is changing the security level session, the DM shall remove all PDIDs from the list of scheduled PDIDs that are not allowed to be read in the new security level.⌋

[SWS_DM_01059] No periodic DIDs in default session

Upstream requirements: RS_Diag_04215

⌈When the diagnostic session changes to or is in the DefaultSession, the DM shall stop all scheduled periodic DIDs.⌋

Scheduler Periodic Transmission

ISO 14229-1[1] defines two distinct scheduler types for ReadDataByPeriodicIdentifier. The DM uses only scheduler type 1 as transmission strategy and NumPeriodicAddr is identical to DiagnosticReadDataByPeriodicIDClass.schedulerMaxNumber. This means that with each scheduler call, all the configured PDIDs for that scheduler rate are transmitted.

[SWS_DM_01060] Support of Scheduler type 1

Upstream requirements: RS_Diag_04215

⌈The DM shall support the scheduling of PDIDs based on scheduler type 1 as defined in ISO 14229-1[1].⌋

[SWS_DM_01061] Trigger all scheduled PDIDs per scheduler

Upstream requirements: RS_Diag_04215

⌈When a periodic scheduler elapses for a requested transmission rate, the DM shall trigger the transmission of all scheduled PDIDs assigned to this transmission rate.⌋

[SWS_DM_01062] Transmission of all PDIDs on the periodic connection

Upstream requirements: RS_Diag_04215

⌈If the PDID transmission is triggered for a requested transmission rate, the DM shall transmit all PDIDs on the periodic transmission on the periodic connection starting with the first PDID in the list of scheduled PDIDs.⌋

[SWS_DM_01063] Transmission error behavior

Upstream requirements: RS_Diag_04215

⌈In case of a PDID transmission error, the DM shall use always the same order of periodicDIDs per client. Transmission errors shall not influence this order, the DM shall continue to retry the transmission and start the next PDID only after the PDID was transmitted successfully.⌋

7.3.2.8.15 Service 0x2C – DynamicallyDefineDataIdentifier

[SWS_DM_01070] Support of UDS service 0x2C in Adaptive AUTOSAR DM

Upstream requirements: RS_Diag_04246

⌈The Diagnostic Server instance shall implement the diagnostic service 0x2C DynamicallyDefineDataIdentifier with subfunctions 0x01 (defineByIdentifier) and 0x03 (clearDynamicallyDefinedDataIdentifier) according to ISO 14229-1[1].⌋

The support of subfunction 0x02 (defineByMemoryAddress) is not appropriate in an AUTOSAR Adaptive system. These systems have mostly more virtual address spaces. Therefore, this subfunction is not supported.

All testers share the same DDDID. In default session the last defined DDDID will be used. The client can switch to a non-default session ensure that no other testers interfere with it's DDDIDs.

[SWS_DM_01071] No persistency of defined DDDIDs

Upstream requirements: RS_Diag_04246

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling is set to volatile, the DM shall initialize all DDDIDs as not present at DM Initialization.⌋

[SWS_DM_01072] Persistency of defined DDDIDs

Upstream requirements: RS_Diag_04246

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling is set to nonVolatile, the DM shall persist DDDIDs configured by the UDS service 0x2C and restore the DDDIDs definition from Non-Volatile Memory at DM Initialization.⌋

[SWS_DM_01073] DM behavior for subfunction 'defineByIdentifier'

Upstream requirements: RS_Diag_04246

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier' the DM shall activate this DDDID and append the associated information received from the diagnostic request: DID source, position and size.⌋

[SWS_DM_01074] Only static DIDs as sourceDataIdentifier

Upstream requirements: RS_Diag_04246

⌈If the DM is receiving the service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier' and one or more of the sourceDataIdentifier are itself a active dynamically defined data identifier, the DM shall return an NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01075] Maximum number of sourceDataIdentifiers in the request

Upstream requirements: RS_Diag_04246

⌈If the DM receives the UDS service 'DynamicallyDefineDataIdentifier' (0x2C) and the number of sourceDataIdentifiers in the request is larger than the configured DiagnosticDynamicallyDefineDataIdentifier.maxSourceElement, the DM shall send NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01076] Clearing all configured DDDIDs

Upstream requirements: RS_Diag_04246

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clearDynamicallyDefinedDataIdentifier and no dynamicallyDefinedDataIdentifier in the request, the DM shall clear all the configured DDDIDs.⌋

[SWS_DM_01077] Clearing individual configured DDDIDs

Upstream requirements: RS_Diag_04246

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clearDynamicallyDefinedDataIdentifier and one dynamicallyDefinedDataIdentifier in the request, the DM shall clear the according DDDID from the request.⌋

A cleared DDDIDs is considered as not configured. A request to read such a DDDID is treated in the same way as a request to a static DID that is not configured.

[SWS_DM_01078] Clear DDDIDs on session change

Upstream requirements: RS_Diag_04246

⌈If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to true and a diagnostic session change occurs, the DM shall clear all defined DDDIDs that have sourceDataIdentifiers that are no longer supported in the new session and security level.⌋

[SWS_DM_01079] Session check for DDDID

Upstream requirements: RS_Diag_04246, RS_Diag_04226

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDDID can be defined in the current session according to [SWS_DM_00413]. If the DDDID cannot be defined in the current session, the DM shall return an NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01080] Security level check for DDDID

Upstream requirements: RS_Diag_04246, RS_Diag_04227

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDDIDs can be defined in the security level according to [SWS_DM_00414]. If the DDDID cannot be defined in the current security level, the DM shall return an NRC 0x33 (Security Access Denied).⌋

[SWS_DM_01081] Session check for sourceDataIdentifier

Upstream requirements: RS_Diag_04246, RS_Diag_04226

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction defineByIdentifier and if DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to true, the DM shall check if the sourceDataIdentifier can be defined in the current session according to [SWS_DM_00413]. If any sourceDataIdentifier cannot be defined in the current session, the DM shall return an NRC 0x31 (requestOutOfRange).⌋

[SWS_DM_01082] Security level check for sourceDataIdentifier

Upstream requirements: RS_Diag_04246, RS_Diag_04227

⌈On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier' and if DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to TRUE, the DM shall check if the sourceDataIdentifier can be defined in the current security level according to [SWS_DM_00414]. If any sourceDataIdentifier cannot be defined in the current security level, the DM shall return an NRC 0x33 (securityAccessDenied).⌋

The DM does further session and security checks for all source DIDs if DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to TRUE when the DDDID is requested with UDS service 0x22 or 0x2A.

[SWS_DM_01083] Use of configured DID ports to get DDDID data

Upstream requirements: RS_Diag_04246

⌈If a DDDID read operation is requested, the DM shall retrieve the data for the dataRecord of the DDDID by using the configuration of the contained DIDs.⌋

7.3.2.8.16 Service 0x2E – WriteDataByIdentifier

The processing of a UDS Service WriteDataByIdentifier (0x2E) is described in ISO 14229-1[1], see in particular the evaluation sequence in Figure 21. On processing, the Diagnostic Server instance needs to perform various checks. The following requirements determine the relation between the input data to be checked and the configuration provided to the Diagnostic Server instance via DEXT parameters.

[SWS_DM_00186] Realisation of UDS service WriteDataByIdentifier (0x2E)

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the diagnostic service 0x2E WriteDataByIdentifier according to ISO 14229-1[1].⌋

[SWS_DM_00415] Check supported DataIdentifier

Upstream requirements: RS_Diag_04203

⌈On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported if and only if there exists a DiagnosticDataIdentifier with id matching the DataIdentifier and this DiagnosticDataIdentifier is referenced by a DiagnosticWriteDataByIdentifier.⌋

[SWS_DM_00416] Check supported DataIdentifier in active session

Upstream requirements: RS_Diag_04203, RS_Diag_04226

⌈On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported in active session if and only if the DataIdentifier is supported according to [SWS_DM_00415] and the active session passes the execution permission check as per [SWS_DM_00101].⌋

[SWS_DM_00417] Check supported DataIdentifier on active security level

Upstream requirements: RS_Diag_04203, RS_Diag_04227

⌈On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported on active security level if and only if the DataIdentifier is supported according to [SWS_DM_00415] and the active security level passes the execution permission check as per [SWS_DM_00103].⌋

[SWS_DM_00572] Writing data for requested DataIdentifier

Upstream requirements: RS_Diag_04097

⌈On reception of the UDS Service WriteDataByIdentifier (0x2E) the Diagnostic Server instance shall provide the data for a DataIdentifier to the mapped RPortPrototypes.⌋

7.3.2.8.17 Service 0x31 – RoutineControl

[SWS_DM_00201] Realization of UDS service RoutineControl (0x31)

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall implement the diagnostic service RoutineControl (0x31) according to ISO 14229-1[1] for subFunctions `startRoutine`, `stopRoutine` and `requestRoutineResults`.⌋

[SWS_DM_00202] Check for Supported RoutineIdentifier and Reaction

Upstream requirements: RS_Diag_04203, RS_Diag_04224

⌈The Diagnostic Server instance shall check, whether the RoutineIdentifier addressed by the UDS Service RoutineControl (0x31) is supported by an existing DiagnosticRoutine with a matching id in the configuration. If the RoutineIdentifier addressed by the UDS Service RoutineControl (0x31) is not supported a negative response with NRC 0x31 (requestOutOfRange) shall be returned.⌋

[SWS_DM_00448] Check supported RoutineIdentifier in active session

Upstream requirements: RS_Diag_04203, RS_Diag_04224, RS_Diag_04226

⌈On reception of the UDS Service RoutineControl (0x31), the DM shall check if the DiagnosticAccessPermission referenced by the DiagnosticRoutine has permissions to be executed in the current session and send a NRC 0x31 if the permissions are not given.⌋

[SWS_DM_00437] Security Level check for RoutineIdentifier

Upstream requirements: RS_Diag_04203, RS_Diag_04224, RS_Diag_04227

⌈On reception of the UDS Service RoutineControl (0x31), the DM shall check if the DiagnosticAccessPermission referenced by the DiagnosticRoutine has permissions to be executed with the current security level and send a NRC 0x33 (Security access denied) if the permissions are not given.⌋

[SWS_DM_00203] Check for Supported Subfunction and Reaction

Upstream requirements: RS_Diag_04203, RS_Diag_04224

⌈The Diagnostic Server instance shall check, whether the Subfunction addressed by the UDS Service RoutineControl (0x31) is supported by checking the existence of the corresponding attributes start or stop or requestResult in the related DiagnosticRoutine configuration. If the Subfunction addressed by the UDS Service RoutineControl (0x31) is not supported by the configuration a negative response NRC 0x12 (SubFunctionNotSupported) shall be returned.⌋

[SWS_DM_00574] UDS Service RoutineControl (0x31) startRoutine processing with generic interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall call the ara::diag::GenericRoutine::Start according to the mapped diagnostic interface [TPS_MANI_01326] and [TPS_MANI_01453] to process the subfunction `startRoutine`.⌋

[SWS_DM_00575] UDS Service RoutineControl (0x31) requestRoutineResults processing with generic interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall call ara::diag::GenericRoutine::RequestResults according to the mapped diagnostic interface [TPS_MANI_01326] and [TPS_MANI_01453] to process the subfunction `requestRoutineResults`.⌋

[SWS_DM_00576] UDS Service RoutineControl (0x31) stopRoutine processing with generic interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall call ara::diag::GenericRoutine::Stop according to the mapped diagnostic interface [TPS_MANI_01326] and [TPS_MANI_01453] to process the subfunction `stopRoutine`.⌋

[SWS_DM_01270] UDS Service RoutineControl (0x31) startRoutine processing with typed interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall utilize the associated RPortPrototype if it is typed by the routineinterfacehierarchicalnamespacelist::routineinterfacename class and call its routineinterfacehierarchicalnamespacelist::routineinterfacename::Start function to process the subfunction `startRoutine`.⌋

[SWS_DM_01271] UDS Service RoutineControl (0x31) stopRoutine processing with typed interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall utilize the associated RPortPrototype if it is typed by the routineinterfacehierarchicalnamespacelist::routineinterfacename class and call its routineinterfacehierarchicalnamespacelist::routineinterfacename::Stop function to process the subfunction stopRoutine.⌋

[SWS_DM_01272] UDS Service RoutineControl (0x31) requestRoutineResults processing with typed interface

Upstream requirements: RS_Diag_04196, RS_Diag_04224

⌈The Diagnostic Server instance shall utilize the associated RPortPrototype if it is typed by the routineinterfacehierarchicalnamespacelist::routineinterfacename class and call its routineinterfacehierarchicalnamespacelist::routineinterfacename::RequestResult function to process the subfunction requestRoutineResults.⌋

7.3.2.8.18 Service 0x34 – RequestDownload

[SWS_DM_00128] Realization of UDS service RequestDownload (0x34)

Upstream requirements: RS_Diag_04196, RS_Diag_04033

⌈The Diagnostic Server instance shall implement the UDS service RequestDownload (0x34) according to ISO 14229-1[1].⌋

[SWS_DM_00867] UDS service RequestDownload (0x34) processing

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall call ara::diag::DownloadService::RequestDownload to process an UDS service RequestDownload (0x34).⌋

7.3.2.8.19 Service 0x35 – RequestUpload

[SWS_DM_00134] Realization of UDS service RequestUpload (0x35)

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the UDS service RequestUpload (0x35) according to ISO 14229-1[1].⌋

[SWS_DM_00868] UDS service RequestUpload (0x35) processing

Upstream requirements: RS_Diag_04033

⌈The Diagnostic Server instance shall call ara::diag::UploadService::RequestUpload to process a UDS service RequestUpload (0x35).⌋

7.3.2.8.20 Service 0x36 – TransferData

[SWS_DM_00137] Realization of UDS service TransferData (0x36)

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the UDS service TransferData (0x36) according to ISO 14229-1[1].⌋

[SWS_DM_00869] UDS service TransferData (0x36) download processing

Upstream requirements: RS_Diag_04033

⌈To download data via the UDS service TransferData, the Diagnostic Server instance shall call ara::diag::DownloadService::DownloadData to process the service.⌋

ISO 14229-1[1] provides a UDS service TransferData (0x36) specific NRC evaluation sequence. This sequence has checks that in rotating order needs to be done by the Diagnostic Server instance and by the service processor itself. Therefore before actually running the service processor, the service processor needs means to do a certain verification step. As the ara::diag::GenericUDSService has only one single method this is not possible for the ara::diag::GenericUDSService. As a result of this, the entire service specific ara::diag::GenericUDSService for UDS service TransferData (0x36).

[SWS_DM_01096] UDS service TransferData (0x36) upload processing

Upstream requirements: RS_Diag_04033

⌈To upload data via the UDS service TransferData, the Diagnostic Server instance shall call ara::diag::UploadService::UploadData to process the service.⌋

Besides the NRC checking according to [SWS_DM_00101] and [SWS_DM_00103] the DM does not further service validation. All other NRC checks required by ISO 14229-1[1] for UDS service TransferData have to be implemented in the application called in [SWS_DM_00869] and [SWS_DM_01096].

7.3.2.8.21 Service 0x37 – RequestTransferExit

[SWS_DM_00141] Realization of UDS service RequestTransferExit (0x37)

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall implement the UDS service RequestTransferExit (0x37) according to ISO 14229-1[1].⌋

[SWS_DM_00871] UDS service RequestTransferExit (0x37) download processing

Upstream requirements: RS_Diag_04033

⌈The Diagnostic Server instance shall call ara::diag::DownloadService::RequestDownloadExit to process a UDS service RequestTransferExit (0x37) for a download.⌋

[SWS_DM_01097] UDS service RequestTransferExit (0x37) upload processing

Upstream requirements: RS_Diag_04033

⌈The Diagnostic Server instance shall call ara::diag::UploadService::RequestUploadExit to process a UDS service RequestTransferExit (0x37) for an upload.⌋

Besides the NRC checking according to [SWS_DM_00101] and [SWS_DM_00103] the DM does not further service validation. All other NRC checks required by ISO 14229-1[1] for UDS service TransferExit have to be implemented in the application called in [SWS_DM_00871] and [SWS_DM_01097].

7.3.2.8.22 Service 0x38 – RequestFileTransfer

This chapter describes the definition and use of the UDS service RequestFileTransfer. The API design had the goal to allow a maximum level of flexibility to efficiently exchange data or even very large data over process bounds in an AUTOSAR Adaptive system. An application developer of the ara::diag::FileTransferService interface can choose between various ways to transfer the data. Some possible ways to do so are:

- Simple: Copy the data between the adaptive application and the Diagnostic Management. This approach scales well enough for smaller file sizes and the implementation is simple.
- Zero Copy: The data is not copied between the adaptive application and the Diagnostic Management. This approach is resources efficient and has a good runtime behavior, especially for larger data transfers.
- OS File System Handle: No data needs to be copied between the adaptive application and the Diagnostic Management. The application provides a file handle

that is used in scope of another process. Data is not copied, but stronger dependencies on the OS exist.

In general, the design is open to implement further strategies that meet the individual project requirements.

[SWS_DM_01310] Realization of UDS service RequestFileTransfer (0x38)

Upstream requirements: RS_Diag_04135

⌈The `Diagnostic Server instance` shall implement the `UDS` service RequestFileTransfer (0x38) according to ISO 14229-1[1]. Besides the `NRC` checking according to [SWS_DM_00101] and [SWS_DM_00103] the `DM` does no further service validation. All other `NRC` checks required by ISO 14229-1[1] for `UDS` service RequestFileTransfer have to be implemented in the adaptive application.⌋

[SWS_DM_01311] Realization of modeOfOperation AddFile (0x01)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to AddFile (0x01) is received, the `Diagnostic Server instance` shall call `ara::diag::FileTransferService::RequestWriteFile` with WriteFileMode equal to kAdd.⌋

[SWS_DM_01312] Realization of modeOfOperation DeleteFile (0x02)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to DeleteFile (0x02) is received, the `Diagnostic Server instance`shall call `ara::diag::FileTransferService::DeleteFile`.⌋

[SWS_DM_01313] Realization of modeOfOperation ReplaceFile (0x03)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to ReplaceFile (0x03) is received, the `Diagnostic Server instance` shall call `ara::diag::FileTransferService::RequestWriteFile` with WriteFileMode equal to kReplace.⌋

[SWS_DM_01314] Realization of modeOfOperation ReadFile (0x04)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to ReadFile (0x04) is received, the `Diagnostic Server instance` shall call `ara::diag::FileTransferService::RequestReadFile`.⌋

[SWS_DM_01315] Realization of modeOfOperation ReadDir (0x05)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to ReadDir (0x05) is received, the `Diagnostic Server instance` shall call `ara::diag::FileTransferService::RequestReadDirectory`.⌋

[SWS_DM_01316] Realization of modeOfOperation ResumeFile (0x06)

Upstream requirements: RS_Diag_04135

⌈If a service RequestFileTransfer with modeOfOperation equal to ResumeFile (0x06) is received, the `Diagnostic Server instance` shall call `ara::diag::FileTransferService::RequestResumeWriteFile`.⌋

The `ara::diag::FileTransferService` methods to process a RequestFileTransfer (0x38) all return a DataTransferReadSession or DataTransferWriteSession object. This object is the connection of processing the RequestFileTransfer (0x38) and the consecutive following TransferData (0x36) services. The 0x36 service that is used to actually transfer the files data stream is done within the `ara::diag::DataTransferReadSession` or `ara::diag::DataTransferWriteSession`. Within these objects various overloaded template constructors exist, that allow that the actual implementation can be inside the handler objects that are part of these template constructors. This allows any custom implementation with different levels of optimization adapted for what is required by the ECU project.

The connection between RequestFileTransfer (0x38) and TransferData (0x36) is done using the corresponding DataTransferReadSession or DataTransferWriteSession provided by a successful processing of the RequestFileTransfer in the `ara::diag::FileTransferService`. This data transfer session is used to abstract between the individual selected strategies to handle the TransferData (0x36) data exchange between the `UDS` protocol and the final destination of the data. This design allows various handler objects to be used. AUTOSAR provides a set of handlers to be choose by the application developer that fits best to the induvial needs (e.g. shared memory or not). The design is also open to provide further handler implementation in future AUTOSAR releases as well as platform vendor specific extensions by adding further template constructors for these handlers in the session objects.

[SWS_DM_01317] Realization of TransferData (0x36) in context of RequestFileTransfer

Upstream requirements: RS_Diag_04135

⌈If a TransferData (0x36) in context of RequestFileTransfer is received, the `Diagnostic Server instance` shall call (depending on the data direction read/write) the corresponding Read or Write methods of the handler classes registered via the template constructors of DataTransferReadSession or DataTransferWriteSession.⌋

[SWS_DM_01318] Realization of RequestTransferExit (0x37) in context of RequestFileTransfer

Upstream requirements: RS_Diag_04135

⌈If a RequestTransferExit (0x37) in context of RequestFileTransfer is received, the Diagnostic Server instance shall call (depending on the data direction read-/write) the Exit method of the handler classes registered via the template constructors of DataTransferReadSession or DataTransferWriteSession.⌋

[SWS_DM_01319] Consecutive registration of notifier with ReleaseHandler::SetNotifier()

Upstream requirements: RS_Diag_04135

⌈In case of a consecutive call of ara::diag::ReleaseHandler::SetNotifier of the corresponding ara::diag::ReleaseHandler instance, the DM shall overwrite the previous registered notifier.⌋

Security and Rights Management

UDS and DEXT allow only a very coarse use of the DiagnosticAccessPermission on service RequestFileTransfer level. A DiagnosticAccessPermission on file level would require a static definition of filenames in DEXT, which is against the nature of a generic file system. This is a difference to the handling of DIDs or RIDs that are defined in DEXT.

As further level a filesystem has a rights management implemented inside the OS and therefore outside the scope of the Diagnostic Server instance. This means that rights management for file access needs to be done in scope of the adaptive application implementing the ara::diag::FileTransferService.

7.3.2.8.23 Service 0x3E – TesterPresent

[SWS_DM_00126] Realisation of UDS service 0x3E TesterPresent

Upstream requirements: RS_Diag_04196

⌈The Diagnostic Server instance shall internally implement the diagnostic service 0x3E TesterPresent according to ISO 14229-1[1].⌋

7.3.2.8.24 Service 0x85 – ControlDTCSetting

The UDS service ControlDTCSetting is used by a client to stop or resume the updating of DTC status bits in the server.

[SWS_DM_00229] Support of UDS service ControlDTCSetting (0x85)

Upstream requirements: RS_Diag_04180, RS_Diag_04159

⌈The Diagnostic Server instance shall provide the UDS service ControlDTCSetting (0x85) according to ISO 14229-1[1].⌋

[SWS_DM_00231] Invalid value for optional request parameter

Upstream requirements: RS_Diag_04203, RS_Diag_04115

⌈If the Diagnostic Server instance receives a UDS service ControlDTCSetting (0x85) request with DTCSettingControlOptionRecord != 0xFFFFFF, the Diagnostic Server instance shall send a NRC 0x31 (RequestOutOfRange).⌋

[SWS_DM_00909] Support of Subfunction 0x01 (ON)

Upstream requirements: RS_Diag_04180, RS_Diag_04159

⌈If the Diagnostic Server instance receives a valid UDS service ControlDTCSetting (0x85) with Subfunction 0x01 (ON) and optionally with DTCSettingControlOptionRecord of value 0xFFFFFF, the Diagnostic Server instance shall:

- enable the update of the UDS DTC status bytes in the event memory
- enable the storage in the event memory
- update ara::diag::ControlDtcStatusType to kDTCSettingOn

⌋

[SWS_DM_00910] Support of Subfunction 0x02 (OFF)

Upstream requirements: RS_Diag_04180, RS_Diag_04159

⌈If the Diagnostic Server instance receives a valid UDS service ControlDTCSetting (0x85) with Subfunction 0x02 (OFF) and optionally with DTCSettingControlOptionRecord of value 0xFFFFFF, the Diagnostic Server instance shall:

- disable the update of the UDS DTC status bytes in the primary event memory and all user defined event memories with DiagnosticMemoryDestinationUserDefined.affectedByControlDTCSetting set to true or not configured
- disable the storage in the primary event memory and all user defined event memories with DiagnosticMemoryDestinationUserDefined.affectedByControlDTCSetting set to true or not configured
- update ara::diag::ControlDtcStatusType to kDTCSettingOff

⌋

[SWS_DM_00811] Re-enabling of ControlDTCSetting by Diagnostic Application

Upstream requirements: RS_Diag_04159, RS_Diag_04211, RS_Diag_04067

⌈In case the DTCSetting is disabled and the Diagnostic Server receives a call to `ara::diag::DTCInformation::EnableControlDtc` function from the application the `Diagnostic Server instance` shall:

- enable the update of the `UDS DTC status byte`
- enable the storage in event memory
- update `ara::diag::ControlDtcStatusType` to `kDTCSettingOn`

⌋

Hint: The monitoring application is responsible for the re-enabling of ControlDTCSetting in case some conditions or states demands so. For this purpose the application can use the interface `ara::diag::DTCInformation` with the function `ara::diag::DTCInformation::EnableControlDtc`.

[SWS_DM_01353] Consecutive registration of notifier with SetControlDtcStatusNotifier()

Upstream requirements: RS_Diag_04159

⌈In case of a consecutive call of `ara::diag::DTCInformation::SetControlDtcStatusNotifier` of the corresponding `ara::diag::DTCInformation` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.2.8.25 Service 0x86 – ResponseOnEvent

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_DM_01978] Supported sub function of ResponseonEvent (0x86) ⌈

Sub function ID	*Sub-function name*	*Kind of sub-function*	*Value of DEXT parameter DiagnosticResponseOnEvent.responseOnEventAction*
0x00	stopResponseOnEvent	Control	`DiagnosticResponseOnEventActionEnum.stop`
0x01	onDTCStatusChange	Setup	`DiagnosticResponseOnEventActionEnum.onDTCStatusChange`

0x03	onChangeOfDataIdentifier	Setup	DiagnosticResponseOnEventActionEnum.onChangeOfDataIdentifier
0x04	reportActivatedEvents	Control	DiagnosticResponseOnEventActionEnum.report
0x05	startResponseOnEvent	Control	DiagnosticResponseOnEventActionEnum.start
0x06	clearResponseOnEvent	Control	DiagnosticResponseOnEventActionEnum.clear
0x07	onComparisonOfValues	Setup	DiagnosticResponseOnEventActionEnum.onComparisonOfValues
0x08	reportMostRecentDtcOnStatusChange	Control	DiagnosticResponseOnEventActionEnum.reportMostRecentDtcOnStatusChange
0x09	reportDTCRecordInformationOnDtcStatusChange	Control	DiagnosticResponseOnEventActionEnum.reportDTCRecordInformationOnDtcStatusChange

⌋

The Diagnostic Server instance supports one active ResponseOnEvent logic per session (the default session or a non-default session) as specified by [1]. In general, the Diagnostic Client that starts ResponseOnEvent owns the whole ResponseOnEvent logic. This means that a Diagnostic Client can take over the ResponseOnEvent logic set up previously by another Diagnostic Client.

[SWS_DM_00491] Realisation of UDS service 0x86 ResponseOnEvent

Upstream requirements: RS_Diag_04160

⌈The Diagnostic Server instance shall implement the diagnostic service 0x86 ResponseOnEvent according to ISO 14229-1:2020 [1].

The implementation rules mentioned in [1], chapter 10.9.1 Service description shall be adhered to.⌋

[SWS_DM_00493] Reestablishing of Client Server communication

Upstream requirements: RS_Diag_04160

⌈In case of a canceled diagnostic conversation this client receives the serviceToRespondTo-responses after a successful reestablishing of a diagnostic conversation.⌋

[SWS_DM_00494] Supported sub functions of ResponseOnEvent service

Upstream requirements: RS_Diag_04160

⌈The `Diagnostic Server instance` shall support the sub functions of diagnostic service 0x86 ResponseOnEvent as listed in [SWS_DM_01978] Supported sub function of Response on Event (0x86).⌋

[SWS_DM_01098] Starting ResponseOnEvent in single and multiple client scenarios

Upstream requirements: RS_Diag_04160

⌈If the ResponseOnEvent sub function startResponseOnEvent is requested by a client, the `Diagnostic Server instance` shall start the ResponseOnEvent logic and send all the serviceToRespondTo messages on the transmission channel of that client.⌋

Note: This means that two or more clients can set up the ResponseOnEvent logic, but only the client that calls startResponseOnEvent receives the messages.

[SWS_DM_01117] Support of eventWindowTime values for ResponseOnEvent

Upstream requirements: RS_Diag_04160

⌈The `Diagnostic Server instance` shall support the event window times as configured by the value of the `DEXT` parameter `DiagnosticResponseOnEvent.eventWindow.eventWindowTime`.⌋

[SWS_DM_01118] Support of `DEXT` parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate

Upstream requirements: RS_Diag_04160

⌈The `Diagnostic Server instance` shall map the `DEXT` parameter `DiagnosticResponseOnEventClass.responseOnEventSchedulerRate` to the ISO 14229-1[1] parameter responseOnEventSchedulerRate.⌋

[SWS_DM_01119] Support of `DEXT` parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentfierEvents

Upstream requirements: RS_Diag_04160

⌈The `Diagnostic Server instance` shall map the `DEXT` parameter `DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentfierEvents` to the ISO 14229-1[1] parameter maxNumChangeOfDataIdentfierEvents.⌋

[SWS_DM_01120] Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents

Upstream requirements: RS_Diag_04160

⌈The Diagnostic Server instance shall map the DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents to the ISO 14229-1[1] parameter MaxNumComparisionOfValueEvents.⌋

[SWS_DM_01121] Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength

Upstream requirements: RS_Diag_04160

⌈The Diagnostic Server instance shall map the DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength to the ISO 14229-1[1] parameter maxSupportedDIDLength.⌋

[SWS_DM_01122] Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumberOfStoredDTCStatusChangedEvents

Upstream requirements: RS_Diag_04160

⌈The Diagnostic Server instance shall map the DEXT parameter DiagnosticResponseOnEventClass.maxNumberOfStoredDTCStatusChangedEvents to the ISO 14229-1[1] parameter maxNumberOfStoredDTCStatusChangedEvents.⌋

[SWS_DM_01262] No storage of RoE events

Upstream requirements: RS_Diag_04160

⌈If DiagnosticResponseOnEventClass.storeEventEnabled is set to FALSE and a RoE service with the storeState bit (Bit6) set to true is received, the DM shall deny the request according to ISO 14229-1[1].⌋

[SWS_DM_01263] Storage of RoE events

Upstream requirements: RS_Diag_04160

⌈If DiagnosticResponseOnEventClass.storeEventEnabled is set to TRUE, the DM shall support the functionaltiy of the RoE service with the storeState bit (Bit6) set to true according to ISO 14229-1[1].⌋

7.3.2.8.26 Custom Diagnostic Services

In addition to supported UDS diagnostic services, as shown in Table 7.2, an OEM or system supplier may have a need for a diagnostic service which is not the part of the services standardized in ISO 14229-1[1].

[SWS_DM_00502] Support for Custom Diagnostic Services

Upstream requirements: RS_Diag_04177

⌈The Diagnostic Server instance shall provide a service processor on SID level for a custom services request (defined by DiagnosticCustomServiceInstance), which is specified by a system supplier.⌋

[SWS_DM_00983] Processing of Custom Diagnostic Services

Upstream requirements: RS_Diag_04177

⌈The Diagnostic Server instance shall call ara::diag::GenericUDSService::HandleMessage to process a custom UDS service by the given SID.⌋

Meta-class DiagnosticCustomServiceInstance can be used to define the instance of a Custom Service. Modeling of Custom Diagnostic Services is described in the TPS Manifest Specification [[11]].

Note: SID range for custom service requests is defined in Table 2 of ISO 14229-1[1].

7.3.2.9 Cancellation of a Diagnostic Conversation

The cancellation of Diagnostic Conversations and external processors is implemented by a CancellationHandler instance. CancellationHandler instance may be part of a service processor implementation also.

The following is the root cause for the cancellation of a Diagnostic Conversation:

- Replacement by a newly requested Diagnostic Conversation according to [SWS_DM_00431].

This section describes the actions to be performed on cancellation of a Diagnostic Conversation.

Cancellation of a Diagnostic Conversation is performed according to the following requirements.

[SWS_DM_00277] Cancellation of a Diagnostic Conversation in case of External Service Processing

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a Diagnostic Conversation in case a diagnostic request is currently processed in context of this Diagnostic Conversation by a service processor external to the Diagnostic Server instance, the Diagnostic Server instance shall notify this external service processor, that the processing for this service shall be canceled according to [SWS_DM_00577].⌋

[SWS_DM_00278] Cancellation of a Diagnostic Conversation in case of Internal Processing

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a `Diagnostic Conversation` in case a diagnostic request is currently processed in context of this protocol internally within the `Diagnostic Server instance`, the `Diagnostic Server instance` shall abort the started activity as far as possible.⌋

[SWS_DM_00279] Cancellation of a Diagnostic Conversation before Response Transmission

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a `Diagnostic Conversation` in case a diagnostic request is currently processed in context of this protocol and response transmission has not yet been started, the `Diagnostic Server instance` shall abort all service processing and skip sending any response, which implies **not** to call `apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit` of the respective UDS Transport Protocol Handler.⌋

[SWS_DM_00280] Cancellation of a Diagnostic Conversation at Response Transmission

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a `Diagnostic Conversation` in case a diagnostic request is currently processed in context of this `Diagnostic Conversation` and `apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit` of the UDS TransportLayer was already called, nothing has to be done by the `Diagnostic Server instance`. This implies a sent out response.⌋

[SWS_DM_00847] Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a `Diagnostic Conversation`, the `Diagnostic Server instance` shall reset the values of the fields of the associated `ara::diag::Conversation` class Instance according to [SWS_DM_00843].⌋

[SWS_DM_00577] Canceling external service processors

Upstream requirements: RS_Diag_04167

⌈On Cancellation of a `Diagnostic Conversation` in case a diagnostic request is currently processed within an external service processor, the supporting `ara::diag::CancellationHandler` shall trigger all notifiers registered via `ara::diag::CancellationHandler::SetNotifier`.⌋

[SWS_DM_00984] Return of cancellation status

Upstream requirements: RS_Diag_04167

⌈A call to `ara::diag::CancellationHandler::IsCanceled` shall return false if the corresponding `Diagnostic Conversation` is not canceled within its current processing state. On Cancellation of this `Diagnostic Conversation` `ara::diag::CancellationHandler::IsCanceled` shall return true until the `ara::diag::CancellationHandler` is destructed.⌋

[SWS_DM_01348] Consecutive registration of notifier with CancellationHandler::SetNotifier()

Upstream requirements: RS_Diag_04167

⌈In case of a consecutive call of `ara::diag::CancellationHandler::SetNotifier` of the corresponding `ara::diag::CancellationHandler` instance, the DM shall overwrite the previous registered notifier.⌋

7.3.3 Diagnostic SOVD Management

7.3.3.1 SOVD Conversations and UDS Interplay

With the introduction of `SOVD` new challenges need to be met regarding parallel client handling. For exclusive client access `UDS` (ISO 14229-1) uses the extended session mechanism and accordingly `SOVD` uses the `SOVD lock` mechanism. Therefore, only one client (`UDS` or `SOVD`) with exclusive client access mechanism (extended session or `SOVD lock`) can have access at a time. For `UDS` clients in default session respectively `SOVD` clients without a lock the same parallel client handling rules as described in section 7.3.2.1.1 apply. Especially since `SOVD` aims on using the same port instances as `UDS` (ISO 14229-1) the same preconditions regarding re-entrant interfaces apply for parallel client access.

[SWS_DM_01477] SOVD lock in UDS extended session

Upstream requirements: RS_Diag_04264

⌈Acquiring a `SOVD lock` shall only be possible if no `UDS` (ISO 14229-1) client is in extended session.⌋

[SWS_DM_01476] Parallel SOVD client handling

Upstream requirements: RS_Diag_04264

⌈The `DM` shall treat a `SOVD` client without a lock like a `UDS` (ISO 14229-1) client regarding parallel client handling.⌋

[SWS_DM_01929] SOVD Error for non-reentrant software

Upstream requirements: RS_Diag_04264

⌈If an SOVD resource is not available due to non-reantrant implementation, the DM shall return the error for "Conflicted state" as per [4].⌋

7.3.3.2 SOVD Request Validation and Verification

7.3.3.2.1 SOVD Authorization

[SWS_DM_01472] Redirection to authorization endpoint

Upstream requirements: RS_Diag_04268

⌈If the SOVD server receives a request for the resource for `Verifying SOVD Client Credentials at the Vehicle` then the DM shall call the method `ara::diag::SovdAuthorization::GetAuthorizationUrl` on the application port mapped via `mapping` to the SoftwareCluster with category PLATFORM and respond with a HTTP status code 307 `Temporary Redirect` and the `Location` header set to the URL that is returned from the method call.⌋

[SWS_DM_01471] Redirection to token endpoint

Upstream requirements: RS_Diag_04268

⌈If the SOVD server receives a request for the resource for `Requesting a Token` then the DM shall call the method `ara::diag::SovdAuthorization::GetTokenUrl` on the application port mapped via `mapping` to the SoftwareCluster with category PLATFORM and respond with a HTTP status code 307 `Temporary Redirect` and the `Location` header set to the URL that is returned from the method call.⌋

application implementer note

The application may evaluate the `queryParameters` and `metaInfo` parameters to redirect the client to the Authentication Server most appropriate for the client.

[SWS_DM_01470] Authorization validation

Upstream requirements: RS_Diag_04268

⌈If an `Authorization` HTTP header is included in the request of the SOVD client and the content (token) is not identical to a token where the validation is still valid, then the `Diagnostic Server instance` shall call the method `ara::diag::SovdAuthorization::ValidateAuthorization` on the application port mapped via `mapping`, passing the contents of the `Authorization` HTTP header of the request as parameter `token`, and a `ara::diag::ClientAuthentication` object.⌋

[SWS_DM_01469] Validity period of authenticated roles

Upstream requirements: RS_Diag_04268

⌈If the call to `ara::diag::SovdAuthorization::ValidateAuthorization` returns a value, then the `Diagnostic Server instance` shall treat any roles that the application has set as authenticated via the `clientAuthentication` parameter as authenticated for any request that is processed no later than the `validUntil` returned from the `ara::diag::SovdAuthorization::ValidateAuthorization` and where the authorization token is identical.⌋

[SWS_DM_01781] (Re-)Identification of Clients by Token

Upstream requirements: RS_Diag_04268

⌈For any subsequent request that is processed no later than the `validUntil` returned from the `ara::diag::SovdAuthorization::ValidateAuthorization` and where the authorization token is identical, the `Diagnostic Server instance` shall consider the client as being identical.⌋

[SWS_DM_01782] (Re-)Identification of Clients by Identity

Upstream requirements: RS_Diag_04268

⌈If the call to `ara::diag::SovdAuthorization::ValidateAuthorization` returns a value that contains an `identity` that is equal to the `identity` of a previous call, then the `Diagnostic Server instance` shall consider the client as being identical.⌋

Note: Being able to re-identify the `SOVD` client is important for locking and access to temporary `SOVD` resources such as operations that were started by the `SOVD` client.

7.3.3.2.2 SOVD Lock

[SWS_DM_01783] Locking only for authorized SOVD clients

Upstream requirements: RS_Diag_04268

⌈`DM` shall only allow a `SOVD` client to acquire a lock on the `SOVD` entity (or `SOVD` sub entity) if the client is successfully authenticated. Otherwise the `DM` shall return HTTP status code 401.⌋

Note: Without an authorization token the `DM` would not be able to re-identify the client.

[SWS_DM_01475] DM shall allow only one lock per SOVD entity

Upstream requirements: RS_Diag_04264

⌈DM shall not allow a SOVD client to acquire a lock on the SOVD entity (or SOVD sub entity) if another SOVD client has locked the same SOVD entity for accessing or operating on the SOVD entity resource and return error code.⌋

[SWS_DM_01474] DM shall allow access only to unlocked SOVD entities

Upstream requirements: RS_Diag_04264

⌈DM shall not allow a SOVD client to access a SOVD entity which requires a lock, but the client has not acquired the lock before, and will send HTTP status code 409 as a response.⌋

[SWS_DM_01473] DM shall lock SOVD entity after time expiration

Upstream requirements: RS_Diag_04264

⌈DM shall terminate all resources associated with the lock as well as temporary resources when expiration time for the acquired lock times out. DM shall return HTTP status code 409 as response when SOVD client tries to further access the deleted resources once the lock times out.⌋

7.3.3.2.3 Validation of Environmental Conditions in SOVD

[SWS_DM_01468] Check of Environmental Conditions before executing SOVD methods

Upstream requirements: RS_Diag_04256, RS_Diag_04273

⌈Before executing a SOVD method, the Diagnostic Server instance shall evaluate the associated Environmental Conditions according to [SWS_DM_00111], [SWS_DM_00112], [SWS_DM_00286], [SWS_DM_00287] and [SWS_DM_00970].⌋

[SWS_DM_01467] Environmental Condition Check for SOVD evaluated to TRUE

Upstream requirements: RS_Diag_04256, RS_Diag_04273

⌈If the evaluation of the Environmental Condition according to [SWS_DM_01468] evaluates to TRUE, the Diagnostic Server instance shall execute the corresponding SOVD method.⌋

[SWS_DM_01466] Environmental Condition Check for SOVD evaluated to FALSE

Upstream requirements: RS_Diag_04256, RS_Diag_04273

⌈If the evaluation of the Environmental Condition according to [SWS_DM_01468] evaluates to FALSE, the Diagnostic Server instance shall not execute the corresponding SOVD method and return status code 503.⌋

7.3.3.2.4 Service Validation of SOVD Requests

[SWS_DM_01784] Service Validation of SOVD Requests

Upstream requirements: RS_Diag_04273

⌈On reception of an SOVD request the DM shall call the according ara::diag::SovdServiceValidation::Validate for each DiagnosticSovdServiceValidationPortMapping configured in DiagnosticServiceValidationConfiguration in the order configured. If any ara::diag::SovdServiceValidation::Validate returns an error, then the DM shall immediately return an error to the client and not proceed processing the request any further.⌋

[SWS_DM_01785] Service Validation for SOVD Resource Collections

Upstream requirements: RS_Diag_04273

⌈If the request is for a resource collection and the service validation of the collection according to [SWS_DM_01784] did not return an error, then for each resource contained in the resource collection that would be returned in the response the DM shall call the according ara::diag::SovdServiceValidation::Validate for each DiagnosticSovdServiceValidationPortMapping configured in DiagnosticServiceValidationConfiguration in the order configured with the resource path in the ara::diag::MetaInfo being set to the path of the contained resource. If any ara::diag::SovdServiceValidation::Validate returns an error any contained resource, then the DM shall immediately return an error to the client and not proceed processing the request any further.⌋

7.3.3.3 SOVD Data Conversion

The exchange of data from a SOVD Client to a SOVD Server and vice versa takes place on an interpreted physical level, whereas internally (as well as for UDS ISO 14220-1 [1]) the data is expressed by internal data types. To express this transformation SwDataDefProps were introduced in Diagnostic Extract Template[3]. This section describes how these transformations shall be interpreted for SOVD and how the subset of a JSON schema that represents a DiagnosticDataElement shall be derived based on Diagnostic Extract Template[3] so that the Diagnostic Server instance can offer consistent ASAM SOVD [4] APIs.

[SWS_DM_01465] SOVD data representation of atomic numeric data

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category IDENTICAL, LINEAR, SCALE_LINEAR, RAT_FUNC or SCALE_RAT_FUNC as defined in Diagnostic Extract Template[3] shall be of type integer or number and

shall derive the JSON schema properties type, minimum, maximum and multipleOf based on baseTypeEncoding, baseTypeSize, dataConstr and compuMethod of the corresponding SwDataDefProps.⌋

[SWS_DM_01464] SOVD data representation of TEXTTABLE

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category TEXTTABLE as defined in Diagnostic Extract Template[3] shall be of type string with JSON property enum. For each compuScale (ordered) one entry of the enum property shall be derived from the vt of the compuConst.⌋

[SWS_DM_01463] SOVD data representation of atomic scaled numeric data with texttable

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category SCALE_LINEAR_AND_TEXTTABLE, SCALE_RAT_AND_TEXTTABLE as defined in Diagnostic Extract Template[3] shall be of type [number, string] or [integer, string], shall derive the JSON schema parameter type, minimum, maximum and multipleOf based on baseTypeEncoding, baseTypeSize, dataConstr, numeric part of the compuMethod and shall derive the JSON schema parameter pattern from the texttable part of the compuMethod.⌋

[SWS_DM_01462] SOVD data representation of TAB_NOINTP

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category TAB_NOINTP as defined in Diagnostic Extract Template[3] shall be of type number or integer with a enum property. For each CompuScale an entry of the enum property shall be derived from the vf of the CompuConst.⌋

[SWS_DM_01461] SOVD data representation of BITFIELD_TEXTTABLE

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category BITFIELD_TEXTTABLE as defined in Diagnostic Extract Template[3] shall be of type object and shall derive one entry of type string with property enum per disjunct interval.⌋

[SWS_DM_01460] SOVD data representation of baseTypeEncoding IEEE754

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding IEEE754 as defined in Diagnostic Extract Template[3] shall be of type number.⌋

[SWS_DM_01459] SOVD data representation of textual Strings

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding of ISO-8859-1, ISO-8859-2, WINDOWS-1252, UTF-8, UTF-16 or UCS-2 as defined in Diagnostic Extract Template[3] shall be of type string and the Diagnostic Server instance shall make the conversion based on the encodings to match the charset defined by ASAM SOVD[4].⌋

[SWS_DM_01458] SOVD data representation of units

Upstream requirements: RS_Diag_04256

⌈The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with a unit referenced as defined in Diagnostic Extract Template[3] shall include the unit in JSON schema according to ASAM SOVD.⌋

[SWS_DM_01457] SOVD data representation of arrays

Upstream requirements: RS_Diag_04256

⌈The subset of a SOVD request and response body matching a DiagnosticDataElement fulfilling [TPS_DEXT_01001] shall be of type array with content type derived accordingly to the atomic data definition. The JSON property maxItems shall be derived from maxNumberOfElements of the DiagnosticDataElement.⌋

7.3.3.4 Standardized APIs

7.3.3.4.1 Docs

Docs is a standardized resource providing an Online Capability description in form of fully specified, self-contained OpenAPI specification file. The description contains information which refers to the creation, reading, updating or deleting of the respective element and its direct child elements as defined by the SOVD standard.

[SWS_DM_01456] SOVD method Query an Online Capability Description

Upstream requirements: RS_Diag_04257

⌈Each SOVD entity shall support the SOVD method Query an Online Capability Description according to ASAM SOVD[4].⌋

[SWS_DM_01786] SOVD Online Capability Description role sensitivity

Upstream requirements: RS_Diag_04257

⌈The response of the SOVD method Query an Online Capability Description shall only contain those SOVD resources that are accessible to the requesting SOVD client.⌋

7.3.3.4.2 Version-Info

The SOVD API uses URI based versioning. SOVD clients can determine which version is supported by an SOVD server by utilization of the resource version-info and querying the respective URI https://<SOVD-Server-Host>/<OEM specific>/version-info.

[SWS_DM_01455] SOVD method for SOVD API Versioning

Upstream requirements: RS_Diag_04256

⌈On SOVD Server level the DM shall support the SOVD method to access SOVD version-info according to ASAM SOVD[4].⌋

[SWS_DM_01454] Query to the SOVD API version

Upstream requirements: RS_Diag_04256

⌈Querying the SOVD API version shall only be supported by SOVD entity SOVD Server.⌋

[SWS_DM_01453] path to version-info

Upstream requirements: RS_Diag_04256

⌈The path to the resource version-info shall remain the same for this and all future versions of the SOVD API.⌋

[SWS_DM_01452] Mismatching versions

Upstream requirements: RS_Diag_04256

⌈If an SOVD client uses a version not supported by the SOVD server, the server shall respond with HTTP status code 404-Not found.⌋

[SWS_DM_01451] SOVDInfo type

Upstream requirements: RS_Diag_04256

⌈The resource version-info shall be of type SOVDInfo.⌋

[SWS_DM_01450] `SOVDInfo` type content

Upstream requirements: RS_Diag_04256

⌈The type `SOVDInfo` shall contain attributes `base_uri` and `version`.⌋

7.3.3.4.3 `Data-categories`

An `SOVD` client can read and write various kinds of data values from and to a `SOVD entity`. Available standard `data-categories` used within data resource collections are identifications, measurements, parameters and system information.

[SWS_DM_01449] `SOVD` method `Retrieve Categories Supported by a Data Resource Collection`

Upstream requirements: RS_Diag_04260, RS_Diag_04258

⌈If a `SOVD entity` support `SOVD` data the `SOVD` method `Retrieve Categories Supported by a Data Resource Collection` shall be support according to `ASAM SOVD`[4].⌋

[SWS_DM_01448] Standard resource `Data categories`

Upstream requirements: RS_Diag_04260

⌈The standardized resource `Data categories` shall be supported by the entities `SOVD` server, Components.⌋

[SWS_DM_01447] `Data categories` type

Upstream requirements: RS_Diag_04260

⌈The resource `data-categories` shall be of type `DataCategory`.⌋

7.3.3.4.4 `Data-groups`

`Data groups` are used to structure data in a specific way (e.g., engine performance related, ...) and are associated with a category.

[SWS_DM_01446] `SOVD` method `Retrieve Groups Supported by a Data Resource Collection`

Upstream requirements: RS_Diag_04260, RS_Diag_04258

⌈If a `SOVD entity` support `SOVD` data and if the `SOVD` has at least one `SOVD` data-group configured, the `SOVD` method `Retrieve Groups Supported by a Data Resource Collection` shall be supported according to `ASAM SOVD`[4].⌋

[SWS_DM_01445] `Data-groups` type

Upstream requirements: RS_Diag_04260

⌈The resource `data-groups` shall be of type `ValueGroup`.⌋

[SWS_DM_01444] `Data-groups` type content

Upstream requirements: RS_Diag_04260

⌈The `ValueGroup` type shall contain attributes `id` and `category`.⌋

7.3.3.4.5 Locks

[SWS_DM_01443] `SOVD` method for locking

Upstream requirements: RS_Diag_04264

⌈The `DM` shall support `SOVD locks` on `Diagnostic Server instance` level according to `ASAM SOVD`[4].⌋

7.3.3.4.6 Logging

[SWS_DM_01928] `SOVD` method for logging

Upstream requirements: RS_Diag_04267

⌈If a `DiagnosticSovdMethod` in context of a `DiagnosticSovdLog` is configured the corresponding `Method for Logging` shall be supported as per `ASAM SOVD`[4].⌋

7.3.3.5 Configurable `APIs`

This section describes how the `Diagnostic Server instance` realizes the configurable `SOVD APIs` as specified in `ASAM SOVD`[4]. Generally there are two different approaches that are used to realize these `SOVD APIs`.

For `SOVD APIs` that have a matching `UDS` Services, the realizing ara::diag interfaces are used for both protocols. Therefore this section aims on describing how the existing configuration patterns are interpreted to realize the corresponding `SOVD APIs`. For example a DiagnosticDataIdentifier with Service ReadByIdentifier (0x22) as of ISO 14229-1[1] is also available as `SOVD data` that supports the `SOVD Read Single Data Value from an Entity API`. This means that the underlying ara::diag interfaces will be used for both protocols. Thus the realizing application will only provide one ara::diag interface, that realizes the `UDS` Service ReadByIdentifier (0x22) as well as

the SOVD Read Single Data Value from an Entity API. The following SOVD methods are realized by this pattern:

- SOVD data
- SOVD configuration (parameter only)
- SOVD faults
- SOVD operations
- Predefined SOVD mode for CommunicationControl and ControlDTCSetting

Unique SOVD APIs, that have no matching UDS Service, are relaized by explicit modeling and by using dedicated ara::diag interfaces. These use-cases are soley These SOVD methods are realized explicelty:

- SOVD bulk-data
- SOVD configuration as parameter and configration as bulk-data
- SOVD software update
- SOVD logs

7.3.3.5.1 Data

[SWS_DM_01442] SOVD method Retrieve List of All Data Provided by the Entity

Upstream requirements: RS_Diag_04260, RS_Diag_04258

⌈If at least one SOVD data is configured, the Diagnostic Server instance shall provide the SOVD method Retrieve List of All Data Provided by the Entity according to ASAM SOVD[4].⌋

[SWS_DM_01441] SOVD data attribute id

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute id of a SOVD data shall be derived from the shortName of the corresponding DiagnosticDataIdentifier.⌋

[SWS_DM_01440] SOVD data attribute name

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute name of a SOVD data shall be derived from the longName of the corresponding DiagnosticDataIdentifier.⌋

[SWS_DM_01439] SOVD data attribute category

Status: DRAFT
Upstream requirements: RS_Diag_04260

⌈The SOVD attribute category of a SOVD data shall be modelled using a SDG on the DiagnosticDataIdentifier with semantics as defined in Listing 7.1.⌋

```
<SDG-DEF>
    <SHORT-NAME>SovdExtensions</SHORT-NAME>
    <SDG-CLASSES>
        <SDG-CLASS>
            <SHORT-NAME>DiagnosticDataIdentifier</SHORT-NAME>
            <GID>sovd:data</GID>
            <EXTENDS-META-CLASS>DiagnosticDataIdentifier</EXTENDS-META-
                CLASS>
            <ATTRIBUTES>
                <SDG-PRIMITIVE-ATTRIBUTE>
                    <SHORT-NAME>sovd_data_category</SHORT-NAME>
                    <CATEGORY>STRING</CATEGORY>
                    <GID>sovd:data_category</GID>
                </SDG-PRIMITIVE-ATTRIBUTE>
            </ATTRIBUTES>
        </SDG-CLASS>
    </SDG-CLASSES>
</SDG-DEF>
<DIAGNOSTIC-DATA-IDENTIFIER>
    <SHORT-NAME>DID</SHORT-NAME>
    <ADMIN-DATA>
        <SDGS>
            <SDG GID="sovd:data">
                <SD GID="sovd:data_category">Measurements</SD>
            </SDG>
        </SDGS>
    </ADMIN-DATA>
</DIAGNOSTIC-DATA-IDENTIFIER>
```

Listing 7.1: SDG class for category of SOVD data. Including SDG-CLASS definition and an example.

[SWS_DM_01437] SOVD group category uniqueness

Upstream requirements: RS_Diag_04260

⌈The aggregated SOVD data within one SOVD group shall all be of the same SOVD category.⌋

[SWS_DM_01436] SOVD group attribute category

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute category of a SOVD group shall be derived by the category of the SOVD data within the SOVD group.⌋

[SWS_DM_01435] SOVD group id

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute id of a SOVD group shall be derived from the shortName of the corresponding collection.⌋

[SWS_DM_01434] SOVD data attribute data of internal structure

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute data of a SOVD data shall be represented by a JSON object literal with one entry per DiagnosticParameter of the corresponding DiagnosticDataIdentifier. The key of each entry shall be derived from the shortName of the corresponding DiagnosticDataElement.⌋

[SWS_DM_01433] SOVD method Read Single Data Value from an Entity

Upstream requirements: RS_Diag_04260

⌈For each SOVD data with Service ReadDataByIdentifier (0x22) the Diagnostic Server instance shall provide the SOVD method Read Single Data Value from an Entity according to ASAM SOVD [4].⌋

[SWS_DM_01432] SOVD method Read Single Data Value from an Entity data query

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute data in the response body of the SOVD method Read Single Data Value from an Entity shall be queried analog to Service ReadDataByIdentifier (0x22) of ISO 14229-1[1] using the same instance of the data querying mechanism.⌋

[SWS_DM_01431] SOVD method Write a Data Value to an Entity

Upstream requirements: RS_Diag_04260

⌈For each SOVD data with Service WriteDataByIdentifier (0x2E) the Diagnostic Server Instance shall provide the SOVD method Write Single Data Value from an Entity according to ASAM SOVD[4].⌋

[SWS_DM_01430] SOVD method Write a Data Value to an Entity data processing

Upstream requirements: RS_Diag_04260

⌈The SOVD attribute data in the request body of the SOVD method Write a Data Value to an Entity shall be processed analog to Service WriteDataByIdentifier (0x2E) of ISO 14229-1[1] using the same instance of the processing mechanism.⌋

7.3.3.5.2 Configuration

The ASAM SOVD[4] standard generally supports two different approaches for the API Methods for Configuration:

- Configurations as parameter using MIME type application/json with defined Response Body structure.
- Configurations as bulk-data using bulk like MIME types such as application/octet-stream

For the realization of these SOVD API Methods for Configuration two different approaches have been realized:

- Configurations as parameter derived from UDS DiagnosticDataIdentifier
- Configurations as parameter or Configurations as bulk-data realized explicitley by modelling and using the dedicated ara::diag::SovdConfiguration::SovdConfiguration Interface

[SWS_DM_01429] SOVD method Retrieve List of All Configurations Provided by the Entity

Upstream requirements: RS_Diag_04261, RS_Diag_04258

⌈If at least one SOVD configuration is configured, the Diagnostic Server instance shall provide the SOVD method Retrieve List of All Configurations Provided by the Entity according to ASAM SOVD[4].⌋

7.3.3.5.2.1 Derived Configurations

[SWS_DM_01428] SOVD configuration attribute id

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute id of a SOVD configuration shall be derived from the shortName of the corresponding DiagnosticDataIdentifier.⌋

[SWS_DM_01427] SOVD configuration attribute name

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute name of a SOVD configuration shall be derived from the longName of the corresponding DiagnosticDataIdentifier.⌋

[SWS_DM_01426] SOVD configuration attribute `type`

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute `type` of a SOVD configuration shall be a constant of type string with value `parameter`.⌋

[SWS_DM_01425] SOVD configuration attribute `version` and `content_type`

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute `version` and `content_type` of a SOVD configuration shall not be supported.⌋

[SWS_DM_01424] SOVD method `Read Configuration as Parameters`

Upstream requirements: RS_Diag_04261

⌈For each SOVD configuration with Service ReadDataByIdentifier (0x22) configured the `Diagnostic Server instance` shall provide the SOVD method `Read Configuration as Parameters` according to ASAM SOVD[4].⌋

[SWS_DM_01423] SOVD method `Read Configuration as Parameters` data query

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute `data` in the response body of the SOVD method `Read Configuration as Parameters` shall be queried analog to Service ReadDataByIdentifier (0x22) of ISO 14229-1[1] using the same instance of the data querying mechanism.⌋

[SWS_DM_01422] SOVD method `Write Configuration as Parameters`

Upstream requirements: RS_Diag_04261

⌈For each SOVD data with Service WriteDataByIdentifier (0x2E) configured the `Diagnostic Server instance` shall provide the SOVD method `Write Configuration as Parameters` according to ASAM SOVD[4].⌋

[SWS_DM_01421] SOVD method `Write Configuration as Parameters` data processing

Upstream requirements: RS_Diag_04261

⌈The SOVD attribute `data` in the request body of the SOVD method `Write Configuration as Parameters` shall be processed analog to Service WriteDataByIdentifier (0x2E) of ISO 14229-1[1] using the same instance of the processing mechanism.⌋

7.3.3.5.2.2 Explicit Configurations

[SWS_DM_01787] Realization of SOVD `Read Configuration`

Upstream requirements: RS_Diag_04261

⌈If the SOVD method `Read Configuration` is called for an explicetly modelled SOVD Configuration the `Diagnostic Server instance` shall call `ara::diag::SovdConfiguration::RequestGetConfiguration`.⌋

[SWS_DM_01788] Realization of SOVD `Write Configuration`

Upstream requirements: RS_Diag_04261

⌈If the SOVD method `Write Configuration` is called for an explicetly modelled SOVD Configuration the `Diagnostic Server instance` shall call `ara::diag::SovdConfiguration::RequestPutConfiguration`.⌋

7.3.3.5.3 `Data-list`

[SWS_DM_01420] SOVD data-list

Upstream requirements: RS_Diag_04260, RS_Diag_04258

⌈The `Diagnostic Server instance` shall provide the SOVD methods `Retrieve List of All Data-Lists Provided by the Entity`, `Creating a Data List for Reading Multiple Data Values at Once from an Entity`, `Read Multiple Data Values at Once from an Entity Using a Data List` and `Delete an Existing Data List` according ASAM SOVD[4].⌋

[SWS_DM_01419] SOVD method `Read Multiple Data Values at Once from an Entity Using a Data List`

Upstream requirements: RS_Diag_04260

⌈The response body for SOVD method `Read Multiple Data Values at Once from an Entity Using a Data List` shall be queried analog to SOVD method `Read Single Data Value from an Entity`.⌋

7.3.3.5.4 Faults

[SWS_DM_01418] SOVD faults

Upstream requirements: RS_Diag_04259

⌈Each `DiagnosticTroubleCodeUds` shall express a SOVD fault.⌋

[SWS_DM_01417] Support of SOVD method `Read Faults from an Entity`

Upstream requirements: RS_Diag_04259, RS_Diag_04258

⌈The `Diagnostic Server instance` shall support the `SOVD` method `Read Faults from an Entity` if a `DiagnosticSovdFaultMemoryAccess` references a `DiagnosticSovdMethod` and the corresponding `DiagnosticSovdMethod.get` attribute exists.⌋

[SWS_DM_01416] Support of SOVD method `Read Details for a Fault`

Upstream requirements: RS_Diag_04259

⌈The `Diagnostic Server instance` shall support the `SOVD` method `Read Details for a Fault` if a `DiagnosticSovdFaultMemoryAccess` references a `DiagnosticSovdMethod` and the corresponding `DiagnosticSovdMethod.get` attribute exists.⌋

[SWS_DM_01415] SOVD fault general attributes

Upstream requirements: RS_Diag_04259

⌈The `SOVD` attributes `code`, `scope`, `display_code`, `fault_name`, `severity` of a `SOVD` fault shall be derived according to [SWS_DM_01567].⌋

[SWS_DM_01567] Response Body for SOVD fault attributes

Upstream requirements: RS_Diag_04259

⌈

SOVD attribute	Deviation rules
code	`udsDtcValue` of `DiagnosticTroubleCodeUds`
scope	`shortName` of corresponding `DiagnosticMemoryDestination`
display_code	`shortName` of `DiagnosticTroubleCodeUds`
fault_name	`longName` of `DiagnosticTroubleCodeUds`
severity	`severity` of `DiagnosticTroubleCodeUds`

⌋

[SWS_DM_01414] SOVD fault attribute `status`

Upstream requirements: RS_Diag_04259

⌈The `SOVD` attribute `status` of a `SOVD` fault shall be a `JSON` object literal with entries according to the `DTC` Status Mask as specified by ISO 14229-1.⌋

[SWS_DM_01413] SOVD fault attribute `symptom`

Status: DRAFT
Upstream requirements: RS_Diag_04259

⌈The SOVD attribute `symptom` of a SOVD fault shall be modelled using a SDG on the `DiagnosticTroubleCodeUds` with semantics as defined in Listing 7.2.⌋

```
<SDG-DEF>
    <SHORT-NAME>SovdExtensions</SHORT-NAME>
    <SDG-CLASSES>
        <SDG-CLASS>
            <SHORT-NAME>DiagnosticTroubleCodeUds</SHORT-NAME>
            <GID>sovd:fault_extensions</GID>
            <EXTENDS-META-CLASS>DiagnosticTroubleCodeUds</EXTENDS-META-
               CLASS>
            <ATTRIBUTES>
                <SDG-PRIMITIVE-ATTRIBUTE>
                    <SHORT-NAME>sovd_fault_symptom</SHORT-NAME>
                    <CATEGORY>STRING</CATEGORY>
                    <GID>sovd:fault_symptom</GID>
                </SDG-PRIMITIVE-ATTRIBUTE>
            </ATTRIBUTES>
        </SDG-CLASS>
    </SDG-CLASSES>
</SDG-DEF>
<DIAGNOSTIC-TROUBLE-CODE-UDS>
    <SHORT-NAME>DTC_UDS</SHORT-NAME>
    <ADMIN-DATA>
        <SDGS>
            <SDG GID="sovd:fault_extensions">
                <SD GID="sovd:fault_symptom">Some generic fault symptom
                   description.</SD>
            </SDG>
        </SDGS>
    </ADMIN-DATA>
</DIAGNOSTIC-TROUBLE-CODE-UDS>
```

Listing 7.2: SDG class for symptoms of SOVD faults. Including SDG-CLASS definition and an example.

[SWS_DM_01412] SOVD fault `environment_data`

Upstream requirements: RS_Diag_04259

⌈The SOVD attribute `environment_data` of a SOVD fault in the response body of the SOVD method `Read Details for a Fault` shall be modelled as described in [SWS_DM_01561].⌋

[SWS_DM_01411] SOVD fault environment_data query

Upstream requirements: RS_Diag_04259

⌈The data for SOVD attribute environment_data of a SOVD fault in the response body of the SOVD method Read Details for a Fault shall be modelled as described in [SWS_DM_01561].⌋

[SWS_DM_01561] Response Body for SOVD fault environment_data table

Upstream requirements: RS_Diag_04259

⌈

SOVD attribute / JSON object parameter key	SOVD Datatype	Deviation Rule
environment_data	object	Available if snapshot record data or extended data is supported for the DTC
environment_data / extended_data_records	object	Available if at least one DiagnosticExtendedDataRecord is configured for the DTC
environment_data / snapshots	array	Available if snapshot record data is configured for the DTC
environment_data / extended_data / {DiagnosticDataRecords.shortName}	object	The key {DiagnosticDataRecords.shortName} shall be derived from the shortName of the corresponding DiagnosticDataRecords
environment_data / extended_data / {DiagnosticDataRecords.shortName} / {DiagnosticDataElement.shortName}		The key {DiagnosticDataElement.shortName} shall be derived from the shortName of the corresponding DiagnosticDataElement. The object shall contain the EDR of the corresponding fault. The content shall be represented analog to the content of the SOVD method Read Single Data Value from an Entity.
environment_data / snapshots / {Array Element}	object	Each freeze frame shall be represented by one element in the array.
environment_data / snapshots / {Array Element} / name	string	In case DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to configured the attribute name shall be derived from the shortName of the corresponding DiagnosticFreezeFrame. In case DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to calculated the attribute name shall be calculated following the pattern "occurence_{occ}", where "{occ}" represents the record numeration. The first occurence shall always use the fix string "first_occurence". The last occurence shall always use the fix string "last_occurence".
environment_data / snapshots / {Array Element} / record_number	integer	The attribute record_number shall be derived according to [SWS_DM_01276].
environment_data / snapshots / {Array Element} / data	object	

SOVD attribute / JSON object parameter key	SOVD Datatype	Deviation Rule
environment_data / snapshots / {Array Element} / data / {DiagnosticDataElement.shortName}		The key {DiagnosticDataElement.shortName} shall be derived from the shortName of the corresponding DiagnosticDataElement. The object shall contain the Daiagnostic Freeze Frame of the corresponding fault. The content shall be represented analog to the content of the SOVD method Read Single Data Value from an Entity.

⌋

[SWS_DM_01409] Support of SOVD method Delete All Faults of an Entity

Upstream requirements: RS_Diag_04259

⌈The Diagnostic Server instance shall support the SOVD method Delete Single Fault of an Entity if a DiagnosticSovdFaultMemoryAccess references a DiagnosticSovdMethod and the corresponding DiagnosticSovdMethod.delete attribute exists.⌋

[SWS_DM_01406] Support of SOVD method Delete Single Fault of an Entity

Upstream requirements: RS_Diag_04259

⌈The Diagnostic Server instance shall support the SOVD method Delete All Faults of an Entity if a DiagnosticSovdFaultMemoryAccess references a DiagnosticSovdMethod and the corresponding DiagnosticSovdMethod.delete attribute exists.⌋

[SWS_DM_02000] Fault memory addressing by the scope parameter ⌈If a SOVD request to read or delete faults with a provided query parameter "scope" is received, the Diagnostic Server instance shall read or delete faults in the fault memory where "scope" is equal to DiagnosticMemoryDestination.shortName.⌋

[SWS_DM_02001] Default fault memory for SOVD requests without scope parameter ⌈If a SOVD request to read or delete faults without query parameter "scope" is received, the Diagnostic Server instance shall read or delete faults in the primary fault memory provided by DiagnosticMemoryDestinationPrimary.⌋

7.3.3.5.5 Operations

[SWS_DM_01405] SOVD operations

Upstream requirements: RS_Diag_04262

⌈Each DiagnosticRoutine shall express a SOVD operation.⌋

[SWS_DM_01404] SOVD method Retrieve List of All Available Operations from an Entity

Upstream requirements: RS_Diag_04262, RS_Diag_04258

⌈If at least one SOVD operation is configured the Diagnostic Server instance shall provide the SOVD method Retrieve List of All Available Operations from an Entity according to ASAM SOVD[4].⌋

[SWS_DM_01403] SOVD method Get Details of a Single Operation

Upstream requirements: RS_Diag_04262

⌈For each SOVD operation configured the Diagnostic Server instance shall provide the SOVD method Get Details of a Single Operation according to ASAM SOVD[4].⌋

[SWS_DM_01402] SOVD operation attribute id

Upstream requirements: RS_Diag_04262

⌈The SOVD attribute id of a SOVD operation shall be derived from the shortName of the corresponding DiagnosticRoutine.⌋

[SWS_DM_01401] SOVD operation attribute name

Upstream requirements: RS_Diag_04262

⌈The SOVD attribute name of a SOVD operation shall be derived from the longName of the corresponding DiagnosticRoutine.⌋

[SWS_DM_01400] SOVD operation attribute proximity_proof_required

Status: DRAFT
Upstream requirements: RS_Diag_04262

⌈The SOVD attribute proximity_proof_required of a SOVD operation shall be modelled using a SDG on the DiagnosticRoutine with semantics as defined in Listing 7.3.⌋

```
<SDG-DEF>
    <SHORT-NAME>SovdExtensions</SHORT-NAME>
    <SDG-CLASSES>
```

```
        <SDG-CLASS>
            <SHORT-NAME>DiagnosticRoutine</SHORT-NAME>
            <GID>sovd:operation_extensions</GID>
            <EXTENDS-META-CLASS>DiagnosticRoutine</EXTENDS-META-CLASS>
            <ATTRIBUTES>
                <SDG-PRIMITIVE-ATTRIBUTE>
                    <SHORT-NAME>sovd_operation_proximity_proof_required</
                        SHORT-NAME>
                    <CATEGORY>BOOLEAN</CATEGORY>
                    <GID>sovd:operation_proximity_proof_required</GID>
                </SDG-PRIMITIVE-ATTRIBUTE>
            </ATTRIBUTES>
        </SDG-CLASS>
    </SDG-CLASSES>
</SDG-DEF>
<DIAGNOSTIC-ROUTINE>
    <SHORT-NAME>Routine</SHORT-NAME>
    <ADMIN-DATA>
        <SDGS>
            <SDG GID="sovd:operation_extensions">
                <SD GID="sovd:operation_proximity_proof_required">TRUE</SD>
            </SDG>
        </SDGS>
    </ADMIN-DATA>
</DIAGNOSTIC-ROUTINE>
```

Listing 7.3: SDG class for proximity_proof_required of SOVD operations. Including SDG-CLASS definition and an example.

[SWS_DM_01399] SOVD operation asynchronous_execution

Upstream requirements: RS_Diag_04262

⌈If the DiagnosticRoutine only has a DiagnosticStartRoutine and no DiagnosticStopRoutine and no DiagnosticRequestRoutineResults configured, the attribute asynchronous_execution of the corresponding SOVD operation shall be false, else the attribute shall be true.⌋

[SWS_DM_01398] SOVD operation proximity_challenge

Upstream requirements: RS_Diag_04262

⌈If the SOVD attribute proximity_proof_required of a SOVD operation is true the Diagnostic Server Instance shall return the attribute proximity_challenge in the response body of SOVD method Get Details of a Single Operation using ara::diag::SovdProximityChallenge::GetChallenge, else the attribute proximity_challenge shall not be supported in the response body.⌋

[SWS_DM_01397] SOVD operation mode

Upstream requirements: RS_Diag_04262

⌈The SOVD attribute modes of a SOVD operation shall not be supported.⌋

[SWS_DM_01396] SOVD method `Start Execution of an Operation`

Upstream requirements: RS_Diag_04262

⌈For each SOVD operation configured the Diagnostic Server Instance shall provide the SOVD method `Start Execution of an Operation` according to ASAM SOVD[4]. The Diagnostic Server Instance shall execute the method analog to Service RoutineControl (0x31) with SubFunction Start (0x01) of ISO 14229-1[1].⌋

[SWS_DM_01395] SOVD method `Start Execution of an Operation proximity_response`

Upstream requirements: RS_Diag_04262

⌈The `proximity_reponse` in the request body of SOVD method `Start Execution of an Operation` shall be validated using `ara::diag::SovdProximityChallenge::GetChallenge`.⌋

[SWS_DM_01394] SOVD method `Get Executions of an Operation`

Upstream requirements: RS_Diag_04262

⌈For each SOVD operation supporting `asynchronous_execution` the `Diagnostic Server instance` shall support the SOVD method `Get Executions of an Operation` according to SOVD SOVD[4].⌋

[SWS_DM_01393] SOVD method `Get the Status of an Operation Execution`

Upstream requirements: RS_Diag_04262

⌈For each `DiagnosticRoutine` with `DiagnosticRequestRoutineResults` the SOVD Server Instance shall provide the SOVD method `Get the Status of an Operation Execution` according to SOVD SOVD[4]. The `Diagnostic Server instance` shall execute the method analog to Service RoutineControl (0x31) with RequestResults (0x03) of ISO 14229-1[1] using the same instance of the processing mechanism.⌋

[SWS_DM_01392] SOVD method `Stop the Execution of an Operation`

Upstream requirements: RS_Diag_04262

⌈For each `DiagnosticRoutine` with `DiagnosticStopRoutine` the SOVD Server Instance shall provide the SOVD method `Stop the Execution of an Operation` according to SOVD SOVD[4]. The `Diagnostic Server instance` shall execute the method analog to Service RoutineControl (0x31) with Stop (0x02) of ISO 14229-1[1] using the same instance of the processing mechanism.⌋

[SWS_DM_01391] SOVD **method** `Stop the Execution of an Operation proximity_response`

Upstream requirements: RS_Diag_04262

⌈The `proximity_response` in the request body of `SOVD` method `Stop the Execution of an Operation` shall be validated using `ara::diag::SovdProximityChallenge::ValidateResponse`.⌋

[SWS_DM_01390] SOVD **operations request and response parameters**

Upstream requirements: RS_Diag_04262

⌈The values for the key parameters in the request body and response body of the `SOVD` methods `Start Execution of an Operation`, `Stop the Execution of an Operation`, `Get the Status of an Operation Execution` shall be represented as `JSON` object literal. The literal shall have one entry per `DiagnosticParameter` with key derived from the `shortName` of the correspoding `DiagnosticDataElement`.⌋

7.3.3.5.6 Modes

[SWS_DM_01389] SOVD **method** `Retrieve List of All Supported Modes of an Entity`

Upstream requirements: RS_Diag_04263, RS_Diag_04258

⌈If at least one `SOVD` mode is configured the Diagnostic Server instance shall provide the `SOVD` method `Retrieve List of All Supported Modes of an Entity` according to `ASAM SOVD`[4].⌋

7.3.3.5.7 `CommunicationControl`

[SWS_DM_01388] SOVD **mode** `commctrl`

Upstream requirements: RS_Diag_04263

⌈If at least one `DiagnosticComControl` is configured, the `Diagnostic Server instance` shall offer a `SOVD` mode `commctrl` with `SOVD` methods `Get Details of a Single Mode of an Entity` and `Explicit Control of Entity States via Their Defined Modes` according to `ASAM SOVD`[4].⌋

[SWS_DM_01387] SOVD mode commctrl name

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute name of the SOVD mode commctrl shall be a constant string with value commctrl.⌋

[SWS_DM_01386] SOVD mode commctrl value schema

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute value of SOVD mode commctrl shall be a JSON enum. For each DiagnosticComControl one possible enum value shall be derived from the category of the corresponding DiagnosticComControl.⌋

[SWS_DM_01385] SOVD mode commctrl get value

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute value in the response body of SOVD method Get Details of a Single Mode of an Entity of SOVD mode commctrl shall return the current state of the Diagnostic Server instance according to Service Communication-Control (0x28) of ISO 14229-1[1].⌋

[SWS_DM_01384] SOVD mode commctrl set value

Upstream requirements: RS_Diag_04263

⌈The SOVD method Explicit Control of Entity States via Their Defined Modes of SOVD mode commctrl with valid value in the request body shall be processed analog to Service CommunicationControl (0x28) of ISO 14229-1[1] using the same instance of the processing mechanism.⌋

7.3.3.5.8 ControlDTCSetting

[SWS_DM_01383] SOVD mode dtcsetting

Upstream requirements: RS_Diag_04263

⌈If a DiagnosticControlDTCSetting is configured, the Diagnostic Server instance shall offer a SOVD mode dtcsetting with SOVD methods Get Details of a Single Mode of an Entity and Explicit Control of Entity States via Their Defined Modes according to ASAM SOVD[4].⌋

[SWS_DM_01382] SOVD mode dtcsetting name

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute name of the SOVD mode dtcsetting shall be a constant string with value dtcsetting.⌋

[SWS_DM_01381] SOVD mode `dtcsetting` value schema

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute `value` of SOVD mode `dtcsetting` shall be a JSON enum with possible values `off` and `on`. These value shall be mapped to the subfunction parameter of Service ControlDTCSetting (0x85) of ISO 14229-1[1].⌋

[SWS_DM_01380] SOVD mode `dtcsetting` get value

Upstream requirements: RS_Diag_04263

⌈The SOVD attribute `value` in the response body of SOVD method `Get Details of a Single Mode of an Entity` of SOVD mode `dtcsetting` shall return the current state of the `Diagnostic Server instance` according to Service ControlDTCSetting (0x85) of ISO 14229-1[1].⌋

[SWS_DM_01379] SOVD mode `dtcsetting` set value

Upstream requirements: RS_Diag_04263

⌈The SOVD method `Explicit Control of Entity States via Their Defined Modes` of SOVD mode `dtcsetting` with valid `value` in the request body shall be processed analog to Service Service ControlDTCSetting (0x85) of ISO 14229-1[1].⌋

7.3.3.5.9 Bulk Data

This section describes realization of the SOVD API Methods for Handling of Bulk Data. This use-case will be mainly handled by the application to match the abstraction of different underlying file systems. `ara::diag::SovdBulkData::SovdBulkData` therefore mainly focuses on handling the underlying SOVD protocol. To allow for maximum flexibility during up- and download, `ara::diag::SovdBulkData::SovdBulkData` uses the data transfer strategies the that have been introduced by UDS Service 0x38 – RequestFileTransfer and are described in detail in section 7.3.2.8.22).

[SWS_DM_01789] Realization of SOVD API Methods for Handling of Bulk Data

Upstream requirements: RS_Diag_04266

⌈The Diagnostic Server instance shall implement the SOVD API Methods for Handling of Bulk Data according to ASAM SOVD [4].⌋

[SWS_DM_01790] Realization of SOVD Bulk Data API Method for Retrieve List of all Bulk Data Categories

Upstream requirements: RS_Diag_04266, RS_Diag_04258

⌈If the SOVD method Bulk Data API Method for Retrieve List of all Bulk Data Categories is called the Diagnostic Server instance shall return all configured Bulk Data Categories according to ASAM SOVD [4].⌋

[SWS_DM_01791] Realization of SOVD Read Bulk Data Meta Data

Upstream requirements: RS_Diag_04266

⌈If the SOVD method Read Bulk Data Meta Data is called the Diagnostic Server instance shall call ara::diag::SovdBulkData::GetBulkDataMetaData.⌋

[SWS_DM_01792] Realization of SOVD Download Bulk Data

Upstream requirements: RS_Diag_04266

⌈If the SOVD method Download Bulk Data is called the Diagnostic Server instance shall call ara::diag::SovdBulkData::RequestBulkDataDownload.⌋

[SWS_DM_01793] Realization of SOVD Upload Bulk Data

Upstream requirements: RS_Diag_04266

⌈If the SOVD method Upload Bulk Data is called the Diagnostic Server instance shall call ara::diag::SovdBulkData::RequestBulkDataUpload.⌋

[SWS_DM_01794] Realization of SOVD Delete All Bulk Data Defined by Category

Upstream requirements: RS_Diag_04266

⌈If the SOVD method Delete All Bulk Data Defined by Category is called the Diagnostic Server instance shall call ara::diag::SovdBulkData::DeleteAllBulkData.⌋

[SWS_DM_01795] Realization of SOVD Delete Specific Bulk Data Resource

Upstream requirements: RS_Diag_04266

⌈If the SOVD method Delete Specific Bulk Data Resource is called the Diagnostic Server instance shall call ara::diag::SovdBulkData::DeleteSpecificBulkData.⌋

7.3.3.5.10 Software Update

This section describes realization of the SOVD API Methods for Software Update. While the ASAM SOVD[4] standard mainly focuses on Software Update on Root/Vehicle Level, it is also possible to use the API on Component level. Figure 7.4 gives an overview on the both SOVD Software Update use cases. This section focuses on the Component Level Software Update APIs.

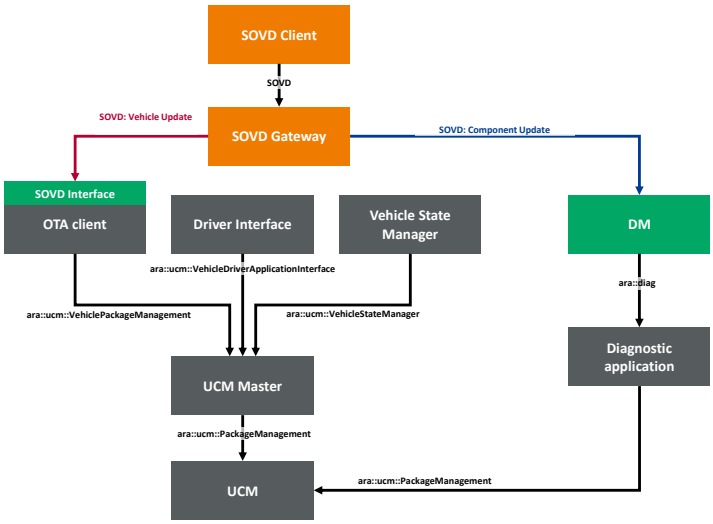

Figure 7.4: SOVD Software Update API - concept. Figure elobarates on the difference of vehicle level and component level SOVD Software Update API.

Since the logic of Software Update is not handled within this Functional Cluster, this use-cases will be mainly handled by the application and ara::diag::SovdSwUpdate::SovdSwUpdate focuses on handling of the underlying SOVD protocol. To allow for maximum flexibility during upload of data, the data transfer strategies that have been introduced by UDS Service 0x38 – RequestFileTransfer will be used. These strategies are described in detail in section 7.3.2.8.22).

[SWS_DM_01796] Realization of SOVD API Methods for Software Update

Upstream requirements: RS_Diag_04265

⌈The Diagnostic Server instance shall implement the SOVD API Methods for Software Update according to ASAM SOVD [4].⌋

[SWS_DM_01797] Realization of SOVD Retrieve List of All Updates

Upstream requirements: RS_Diag_04265, RS_Diag_04258

⌈If the SOVD method Retrieve List of All Updates is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::GetAllUpdates.⌋

[SWS_DM_01798] Realization of SOVD Get Details of Update

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Get Details of Update is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::GetUpdatePackageDetails.⌋

[SWS_DM_01799] Realization of SOVD Automated Installation of an Update

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Automated Installation of an Update is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::PutUpdatePackageAutomated.⌋

[SWS_DM_01800] Realization of SOVD Prepare Installation of an Update

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Prepare Installation of an Update is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::PrepareUpdatePackage.⌋

[SWS_DM_01801] Realization of SOVD Execute Installation of an Update

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Execute Installation of an Update is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::ExecuteUpdatePackage.⌋

[SWS_DM_01802] Realization of SOVD Get Status of an Update

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Get Status of an Update is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::GetUpdatePackageStatus.⌋

[SWS_DM_01803] Realization of SOVD Delete Update Package from an SOVD Server

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Delete Update Package from an SOVD Server is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::DeleteUpdatePackage.⌋

[SWS_DM_01804] Realization of SOVD Register an Update at the SOVD Server

Upstream requirements: RS_Diag_04265

⌈If the SOVD method Register an Update at the SOVD Server is called the Diagnostic Server instance shall call ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration.⌋

7.3.4 Diagnostic Event Management

7.3.4.1 Diagnostic Events

7.3.4.1.1 Event Definition

Diagnostic events are used by applications to report the state of a monitored entity to the DM. An event uniquely identifies the monitored entity in the system. The DM receives event notifications from the applications and performs defined actions such as DTC status changes or capturing and storage of extended data records or snapshot records. In other words, events are the input source for the Diagnostic Event Management unit of the DM.

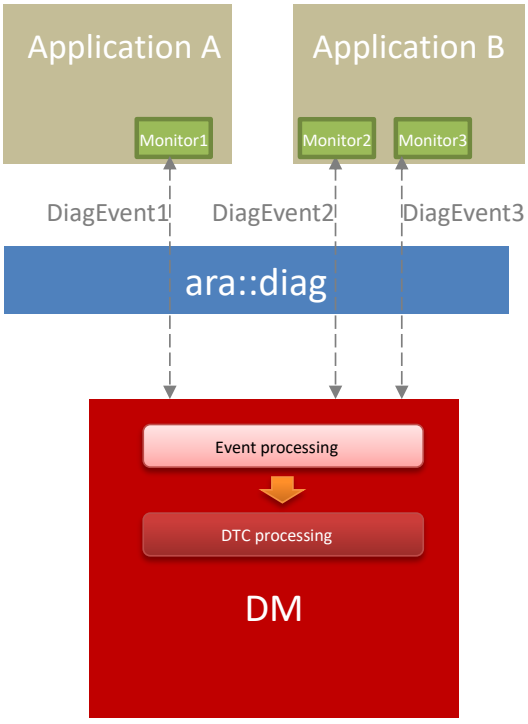

Figure 7.5: Example of diagnostic event usage

An event represents a specific diagnostic monitoring performed which is unique within the system and DM only supports notifications for a certain event from one single

source. This implies that only one application can report a certain event and the event reporting interface is explicitly not re-entrant.

The available events are derived from DiagnosticEvent.

The DM provides two types of interfaces that can be used from an AA to read information for an event. The ara::diag::Event interface provides operations that are assigned for one single event, that is mapped via DiagnosticEventPortMapping. For each DiagnosticEvent the AA needs to instantiate exactly one ara::diag::Event instance. The ara::diag::MultipleEvent provides the same functionality, but there is one interface that can be used for multiple DiagnosticEvent elements. This is helpful for applications that are handling many events and the creation of a large amount of ara::diag::Event interfaces would be an overhead. In that case a single interface can be used for many events and each operation provides a further eventHandle as parameter to identify the event for which the operation is applicable. Each DiagnosticEvent requires a mapping via DiagnosticMultipleEventPortMapping to express, that a multiple event interface can be used for operation on this event.

Function wise both interfaces offer the same kind of operations and do have the same semantics. For simplicity in this document the functionality of the event class is described only via ara::diag::Event. Whenever a functionality of the ara::diag::Event class is described, the same applies for the operation on the ara::diag::MultipleEvent.

[SWS_DM_01733] MultipleEvent EventHandleType Generation

Upstream requirements: RS_Diag_04165, RS_Diag_04201

⌈The EventHandleType is a numerical value created by the following rules:

- If no DiagnosticMultipleEventPortMapping.overrideId exists, in alphabetical order of DiagnosticEvent shortname the DM assigns for the first DiagnosticEvent a number of 0x010000 and increments this number for each further DiagnosticEvent.
- If DiagnosticMultipleEventPortMapping.overrideId exists, the number idOverride shall be used instead.
- If an automatic created number collides with a preconfigured idOverride value, the next free value shall be chosen.

⌋

The idOverride is used to allow the application developer or integrator to use/reuse a fixed number from the application that is not changed if the auto generated number changes due to an update of the input configuration with new events or changed names. The numbers are greater or equal to 0x010000, reserving the range between 0x0000 and 0xFFFF for future extensions and definitions.

[SWS_DM_01734] Invalid EventHandleType

Upstream requirements: RS_Diag_04105

⌈If any ara::diag::MultipleEvent method with input parameter eventHandle is called and this instance of MultipleEvent has not mapped DiagnosticMultipleEventPortMapping that would create the provided EventHandleType, the DM shall return kInvalidArgument.⌋

7.3.4.1.2 Monitors

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

In general diagnostic monitors are independent from the DM. Once the ECU is started and initialized they are permanently monitoring a part of the system and reporting the state to the DM. There are use cases, where it might not be required to continue to monitor the system part and the monitor could stop its task until a certain situation arises.

The ara::diag::Monitor interface is used by the DM to provide applications the ability to report monitor states to the DM. From this perspective the ara::diag::Monitor class is a required port. This class also has a notifier that is called by the DM to inform the AA about monitor state changes. From this perspective the interface is a provided port. This makes the ara::diag::Monitor interface different to other classes and this is also the reason that ara::diag::Monitor has an ara::diag::Monitor::Offer and ara::diag::Monitor::StopOffer method that usually is only available for provided ports.

The ara::diag::Monitor::Offer method is necessary to provide an application the possibility to inform the DM that the application is ready to receive calls on the registered notifier. The notifier is registered in the constructor of the ara::diag::Monitor class. The concrete instance of the monitor object is only known after the object is created. As monitor class and the notifier instance are coupled (the notifier needs to know for which monitor it is used) the application needs to tell its notifier for which monitor object it is created. Therefore after creation of the ara::diag::Monitor class is not active until ara::diag::Monitor::Offer is called. Before ara::diag::Monitor::Offer is called, the application can complete the initialization of the notifier by providing the notifier implementation with a reference to the monitor object. Without this, the DM could call the notifier immediately after the object is constructed into a notifier object, that would not be fully initialized from application point of view.

The ara::diag::Monitor::StopOffer is the counterpart of the ara::diag::Monitor::Offer. At shutdown it defines a point in time where the application requests the monitor object to be no longer used and after ara::diag::Monitor::StopOffer it no longer expects calls on the notifier. After ara::diag::Monitor::StopOffer there are also no further calls on the corresponding ara::diag::Event::GetFaultDetectionCounter method. Without the ara::diag::Monitor::StopOffer it would be undefined until when an application will receive calls on the notifier and the notifier object could be safely discarded.

To avoid these kind of race conditions in object initialization/deinitialization the ara::diag::Monitor::Offer and ara::diag::Monitor::StopOffer methods are introduced. The names ara::diag::Monitor::Offer and ara::diag::Monitor::StopOffer are chosen to be the same names as the ara::com Offer() and StopOffer() as they have basically the same function. It would be of cause possible to name these methods like Activate() and Deactivate() but this would only introduce new names for exactly the same. Therefore AUTOSAR has chosen the ara::diag::Monitor::Offer and ara::diag::Monitor::StopOffer and it is emphasized that this is a diag::Monitor specific implementation not to be mixed with ara::com Offer() and StopOffer().

Besides the reporting direction of the monitors (AAs report the monitoring status towards the DM), there is also a connection in the opposite direction: The DM uses the

- initMonitor (for Monitors with Monitor-internal debouncing)
- initMonitor (for Monitors with counter-based debouncing)
- initMonitor (for Monitors with time-based debouncing)

notifier to trigger a (re-)initialization of diagnostic monitors in the AA. The above listed notifier methods are optional and can be registered by one of the ara::diag::Monitor constructors.

[SWS_DM_01740] Initial enable condition state with notifier callback for disabled enable conditions

Upstream requirements: RS_Diag_04270

⌈If ara::diag::Monitor::Offer is called and the ara::diag::Monitor has a registered initMonitor callback and the enabled condition assigned to that event is in state kDisabled, the DM shall call the registered notifier method ara::diag::Monitor::Monitor::initMonitor with the parameter ara::diag::InitMonitorReason set to kDisabled.⌋

[SWS_DM_00562] Monitor initialization for clearing reason

Upstream requirements: RS_Diag_04185

⌈If an associated DTC, belonging to the current monitoring path, was actually cleared, the DM shall call the registered notifier method ara::diag::Monitor::Monitor::initMonitor with the parameter ara::diag::InitMonitorReason set to kClear.⌋

[SWS_DM_00563] Monitor initialization for operation cycle restart reason

Upstream requirements: RS_Diag_04186

⌈If a diagnostic event was (re)started by calling the method ara::diag::OperationCycle::RestartOperationCycle, the DM shall call the registered notifier method ara::diag::Monitor::Monitor::initMonitor with the parameter ara::diag::InitMonitorReason set to kRestart.⌋

[SWS_DM_01094] Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On

Upstream requirements: RS_Diag_04125, RS_Diag_04159, RS_Diag_04270

⌈In case an enable condition mapped to the diagnostic event was changed to fulfilled and in this way all related enable conditions of the event were fulfilled and DTC-setting is re-enabled via the UDS service request ControlDTCSetting - 0x85 set to On (see ISO 14229-1[1]), the DM shall call the registered notifier method ara::diag::Monitor::Monitor::initMonitor with the parameter ara::diag::InitMonitorReason set to kReenabled.⌋

[SWS_DM_01095] Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off

Upstream requirements: RS_Diag_04125, RS_Diag_04192, RS_Diag_04159

⌈In case an enable condition mapped to the diagnostic event was changed to not fulfilled or DTC-setting is disabled via the UDS service request ControlDTCSetting - 0x85 set to Off (see ISO 14229-1[1]), the DM shall call the registered notifier method ara::diag::Monitor::Monitor::initMonitor with the parameter ara::diag::InitMonitorReason set to kDisabled.⌋

Per diagnostic monitor an instance of the class ara::diag::Monitor is created by the application. Diagnostic results are reported to the DM via the method ara::diag::Monitor::ReportMonitorAction. The method ara::diag::Monitor::ReportMonitorAction calculates the update of the corresponding instance of ara::diag::Event (from DiagnosticEvent) and the UDS DTC status byte as well as the storage in the event memory and the capturing of DTC related data.

The connection between the ara::diag::Monitor interface and the DM might be not active, when ara::diag::Monitor::ReportMonitorAction is called. A typical situation is the start up phase of the ECU, the DM process might be up and running after the application is started. In that case, the DM provides means to cache the action provided with ara::diag::Monitor::ReportMonitorAction to bridge the time-span until the DM is up and running and the connection between ara::diag::Monitor and the DM is established. Also in case the connection is lost and reestablished, the caching is used.

[SWS_DM_00965] Caching of monitor results

Upstream requirements: RS_Diag_04179

⌈If the function ara::diag::Monitor::ReportMonitorAction is called and the DM is currently not ready to process reported event qualification, the DM shall cache at least one qualified PASSED and qualified FAILED result and process the qualified FAILED/PASSED, when the daemon connection is (re-)established.⌋

Note: The debouncing is executed independently of this caching. Only the event qualification trigger is stored in the cache (e.g. no snapshot data included).

The DM provides two types of interfaces that can be used from an AA to report monitor action to the DM. The ara::diag::Monitor interface provides operations that are assigned for one single monitor, that is mapped via DiagnosticMonitorPortMapping. For each DiagnosticEvent the AA needs to instantiate exactly one ara::diag::Monitor instance. The ara::diag::MultipleMonitor provides the same functionality, but there is one interface that can be used for multiple DiagnosticEvent elements. This is helpful for applications that are handling monitors for many events and the creation of a large amount of ara::diag::Monitor interfaces would be an overhead. In that case a single interface can be used for many monitors and each operation provides a further monitorHandle as parameter to identify the monitor for which the operation is applicable. Each DiagnosticEvent requires a mapping via DiagnosticMultipleMonitorPortMapping to express, that a multiple monitor interface can be used for operation on this monitor.

Function wise both interfaces offer the same kind of operations and do have the same semantics. For simplicity in this document the functionality of the monitor class is described only via ara::diag::Monitor. Whenever a functionality of the ara::diag::Monitor class is described, the same applies for the operation on the ara::diag::MultipleMonitor.

There is a minor difference in selection of the debounce algorithms used for an event. For ara::diag::Monitor the debounce algorithms are selected within the constructor. For ara::diag::MultipleMonitor this is not possible, as only one object constructor is available, but the individual events can have different debounce settings. Therefore, this interface provides the overloaded operation InitDebouncing. Function wise this will set the used debounce algorithms in the same way as it is done in the constructor for a ara::diag::Monitor.

[SWS_DM_01731] MultipleMonitor MonitorHandleType Generation

Upstream requirements: RS_Diag_04063

⌈The MonitorHandleType is a numerical value created by the following rules:

- If no DiagnosticMultipleMonitorPortMapping.overrideId exists, in alphabetical order of DiagnosticEvent shortname the DM assigns for the first DiagnosticEvent a number of 0x010000 and increments this number for each further DiagnosticEvent.

- If `DiagnosticMultipleMonitorPortMapping.overrideId` exists, the number idOverride shall be used instead.
- If an automatic created number collides with a preconfigured idOverride value, the next free value shall be chosen.

⌋

The idOverride is used to allow the application developer or integrator to use/reuse a fixed number from the application that is not changed if the auto generated number changes due to an update of the input configuration with new events or changed names. The numbers are greater or equal to 0x010000, reserving the range between 0x0000 and 0xFFFF for future extensions and definitions.

[SWS_DM_01732] Invalid MonitorHandleType

Upstream requirements: RS_Diag_04179, RS_Diag_04063

⌈If any `ara::diag::MultipleMonitor` method except of `action` with input parameter monitorHandle is called and this instance of MultipleMonitor has not mapped `DiagnosticMultipleMonitorPortMapping` that would create the provided MonitorHandleType, the `DM` shall return `kInvalidArgument`.⌋

Some monitor actions are not supported by given debouncing methods. The following table shows for each debouncing method which monitor action is supported and which monitor action reporting will lead to a violation.

[SWS_DM_01987] Supported debouncing method for which monitor action ⌈

...	*Debouncing method*		
MonitorAction	***Counter-based***	***Timer-based***	***Internal debouncing***
kPassed	supported	supported	supported
kFailed	supported	supported	supported
kPrepassed	supported	supported	not supported
kPrefailed	supported	supported	not supported
kFdcThresholdReached	not supported	not supported	supported
kResetTestFailed	supported	supported	supported
kFreezeDebouncing	not supported	supported	not supported
kResetDebouncing	supported	supported	not supported

⌋

[SWS_DM_01988] Unexpected monitor action handling ⌈If `ara::diag::Monitor::ReportMonitorAction` is called with a parameter "`action`" value that is not supported by the used debouncing method as per the [SWS_DM_01987], the `DM` shall trigger a violation according to [SWS_CORE_00003] in the process of the corresponding `ara::diag::Monitor` interface.⌋

7.3.4.1.3 Event Combination

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

The `DM` supports the following event combination types:

- Combination on storage: The combined `DTC` is stored and updated in a single `event memory` entry. Event related data is stored only for one `event`.
- Combination on retrieval: Each `event` is stored in a separate `event memory` location. The event related data for all stored `events` are reported for a `DTC`.

Event combination is enabled by setting `DiagnosticCommonProps.typeOfEventCombinationSupported` to one of the supported event combination types. Event combination types have a direct impact on `UDS` response messages and therefore are available individually per `event memory`.

[SWS_DM_01106] Applicability of Event Combination

Upstream requirements: RS_Diag_04200

⌈The `DM` shall apply event combination for a `DTC` if more than one `DiagnosticEventToTroubleCodeUdsMapping` refer the same `DiagnosticTroubleCodeUds`.⌋

[SWS_DM_01976] shows an example configuration with event combination in place. Several `events` are mapped to the same `DTC`. The combination on storage is represented by the DTC[1]. The DTC[2] shows an example of the combination on retrieval. The freeze frame data is stored individually for each `event`.

[SWS_DM_01976] Example of a configuration table including combined DTCs ⌈

Unique EventId	*Monitor status*	*UDS status*	*Assigned DTC*	*Freeze frame*	*....*
Event[1]	S1	S1|S2|S3	DTC[1]	FF[28]	
Event[2]	S2	S1|S2|S3	DTC[1]	FF[28]	
Event[3]	S3	S1|S2|S3	DTC[1]	FF[28]	
Event[4]	S4	S4|S5|S6	DTC[2]	FF[74]	
Event[5]	S5	S4|S5|S6	DTC[2]	FF[77]	

Event[6]	S6	S4|S5|S6	DTC[2]	FF[75]	
Event[7]	S7	S7	DTC[3]	FF[89]	
Event[8]	S8	S8	DTC[4]	FF[67]	
....	....	....	....	....	....

⌋

[SWS_DM_01107] DTC Status Byte calculation

Upstream requirements: RS_Diag_04200

⌈If `DiagnosticCommonProps.typeOfEventCombinationSupported` is configured, the `DM` shall calculate the `UDS DTC status byte` according to [SWS_DM_01977] for all `events` with event combination that are mapped to `DTCs` in that `DiagnosticCommonProps`.⌋

The `ara::diag::EventStatusBit` type does not define all `UDS DTC status bits` for an `Event`. Some of the `Event status` bits for the calculation in [SWS_DM_01977] need to be derived by the Diagnostic Manager. The Derived Event Status of an individual event will be calculated as if it were a `UDS DTC status byte` as specified in 7.3.4.4.2 "UDS DTC Status". The Derived `event status` is not available outside of the Diagnostic Manager.

[SWS_DM_01977] Calculation of UDS status byte ⌈

UDS status bit description		***Combined UDS status information logical equation***
0	TestFailed	$CbDTC_{Bit0}$ = $Event[1]_{Bit0}$ | $Event[2]_{Bit0}$ | ... |$Event[n]_{Bit0}$
1	TestFailedThis OperationCycle	$CbDTC_{Bit1}$ = $Event[1]_{Bit1}$ | $Event[2]_{Bit1}$ | ... |$Event[n]_{Bit1}$
2	PendingDTC	$CbDTC_{Bit2}$ = $Event[1]_{Bit2}$ | $Event[2]_{Bit2}$ | ... |$Event[n]_{Bit2}$
3	ConfirmedDTC	$CbDTC_{Bit3}$ = $Event[1]_{Bit3}$ | $Event[2]_{Bit3}$ | ... |$Event[n]_{Bit3}$
4	TestNotCompleted SinceLastClear	$CbDTC_{Bit4}$ = ($Event[1]_{Bit4}$ | $Event[2]_{Bit4}$ | ... |$Event[n]_{Bit4}$) & ! $CbDTC_{Bit5}$
5	TestFailedSince LastClear	$CbDTC_{Bit5}$ = $Event[1]_{Bit5}$ | $Event[2]_{Bit5}$ | ... |$Event[n]_{Bit5}$
6	TestNotCompleted ThisOperationCycle	$CbDTC_{Bit6}$ = ($Event[1]_{Bit6}$ | $Event[2]_{Bit6}$ | ... |$Event[n]_{Bit6}$) & ! $CbDTC_{Bit1}$CbDTC
7	WarningIndicator Requested	$CbDTC_{Bit7}$ = $Event[1]_{Bit7}$ | $Event[2]_{Bit7}$ | ... |$Event[n]_{Bit7}$

⌋

In table [SWS_DM_01977] the following logical operators are used:

- ! = logical negation (NOT)

- | = logical bitwise OR-operation
- & logical bitwise AND-operation

The calculation of the EventStatusByte is not affected by any event combination (i.e. EventStatusByte always represents the status of the corresponding event).

Event combination has no effect on clearing DTCs.

[SWS_DM_01108] Clear all DTCs event with event combination

Upstream requirements: RS_Diag_04200

⌈If the DM is performing a clearDTC operation, the DM shall clear all related events independently if the event combination is used or not.⌋

[SWS_DM_01109] UDS DTC status update for combined DTCs

Upstream requirements: RS_Diag_04200

⌈Each time the status of an event is updated, the DM shall calculate the combined DTC status.⌋

[SWS_DM_01110] Callbacks for combined UDS DTC status change

Upstream requirements: RS_Diag_04200

⌈Each time the combined DTC status has changed, the DM shall call all configured callbacks.⌋

The DTC fault detection counter is a signed value in the range between -128 and +127. A positive value represents the situation that an event has been reported as failed debouncing of a qualified failed is in place. For event combination the scenario that an event is about to be qualified failed gets a higher priority than the scenario that an event is about to be qualified passed.

[SWS_DM_01111] Fault detection counter for combined events

Upstream requirements: RS_Diag_04200

⌈If event combination is used for the events of an DTC, the DM shall calculate the fault detection counter (FDC) of that DTC by taking the greatest fault detection counter value of the sub-events.⌋

[SWS_DM_01269] Requesting DTC number for events without DTC

Upstream requirements: RS_Diag_04200

⌈If ara::diag::Event::GetDTCNumber is called and no DiagnosticEvent-ToTroubleCodeUdsMapping exists between the event and UDS DTC, the DM shall return the error code kNoSuchDTC.⌋

7.3.4.1.3.1 Combination On Storage

For event combination on storage, the DM stores event related data only for one single event per DTC. The reporting of the event related data via UDS behaves in the same way as only one event would be assigned for this DTC.

[SWS_DM_01112] Event memory entry for events with the combination on storage

Upstream requirements: RS_Diag_04200

⌈If DiagnosticCommonProps.typeOfEventCombinationSupported is set to eventCombinationOnStorage, then, the DM shall use the combined UDS DTC status bit transitions as trigger source for the allocation of the event memory entry, collection or update of the event related data.⌋

[SWS_DM_01113] Aging counter for combined events

Upstream requirements: RS_Diag_04200

⌈If DiagnosticCommonProps.typeOfEventCombinationSupported is set to eventCombinationOnStorage, the DM shall calculate the aging counter based on the combined DTC status.⌋

7.3.4.1.3.2 Combination On Retrieval

For event combination on retrieval, the DM allocates the stored data per event in the same way as if no event combination is used. The event combination is done on data retrieval while reporting the event related data via UDS. The positive response message for ReadDTCInformation contains the data of all stored events concatenated. Thus, the length of the response message increases with each event that has stored data for this DTC.

[SWS_DM_01114] Data storage for event combination on retrieval

Upstream requirements: RS_Diag_04200, RS_Diag_04067

⌈If DiagnosticCommonProps.typeOfEventCombinationSupported is set to eventCombinationOnRetrieval, the DM shall trigger the collection, update and storage of event related data per event.⌋

[SWS_DM_01115] Data reporting for event combination on retrieval

Upstream requirements: RS_Diag_04200, RS_Diag_04067

⌈If DiagnosticCommonProps.typeOfEventCombinationSupported is set to eventCombinationOnRetrieval, the DM shall return the event related data of all

assigned events to UDS by concatenating the data of all events into one response for the same DTC.⌋

The aging and displacement of DTC with event combination on retrieval is done in the same way as for DTCs with only one assigned event.

With event combination on retrieval the positive response of a UDS request to retrieve snapshot records or extended data records contains the same record multiple times. In [SWS_DM_01116] there is an optional definition for a reporting order. Chronological order is based on the time the event was initially stored in the server. The event storage is not the storage of the record, as an event entry store multiple records and only a chronological order of events not of the individual records is provided.

[SWS_DM_01116] Reporting order of snapshot and extended data records

Upstream requirements: RS_Diag_04200

⌈If DiagnosticCommonProps.typeOfEventCombinationSupported is set to eventCombinationOnRetrieval and DiagnosticCommonProps.eventCombinationReportingBehavior is set to reportingInChronlogicalOrderOldestFirst and the DM is reporting snapshot records or extended data records and for one record number multiple events have data, the DM shall report the stored event data in chronological order of the event data storage with the oldest entry first.⌋

7.3.4.1.4 EnableConditions

The DM provides also Enable Conditions to ignore a certain call of ara::diag::Monitor::ReportMonitorAction in required situations.

DiagnosticEnableConditions are mapped to DiagnosticEvents by DiagnosticEventToEnableConditionGroupMappings.

Enable Conditions are controlled via the function ara::diag::Condition::SetCondition. There current corresponding status, either ' kConditionFalse' or kConditionTrue can be retrieved via 'ara::diag::Condition::GetCondition'.

[SWS_DM_00568] Handling of enable conditions

Upstream requirements: RS_Diag_04192

⌈If the function ara::diag::Monitor::ReportMonitorAction is called and one or more of the event assigned enable conditions are not fulfilled, the DM shall ignore the reported monitor result.⌋

Note: For a regular processing of ara::diag::Monitor::ReportMonitorAction **all** of the enable conditions mapped to the corresponding event have to be fulfilled.

The DM provides two types of interfaces that can be used from an AA to set or get conditionstates. The ara::diag::Condition interface provides operations that are assigned for one single condition, that is mapped via DiagnosticEnableConditionPortMapping. For each DiagnosticCondition the AA needs to instantiate exactly one ara::diag::Condition instance. The ara::diag::MultipleCondition provides the same functionality, but there is one interface that can be used for multiple DiagnosticCondition elements. This is helpful for applications that are handling many conditions and the creation of a large amount of ara::diag::Condition interfaces would be an overhead. In that case a single interface can be used for many conditions and each operation provides a further conditionHandle as parameter to identify the condition for which the operation is applicable. Each DiagnosticCondition requires a mapping via DiagnosticMultipleConditionPortMapping to express, that a multiple condition interface can be used for operation on this condition.

Function wise both interfaces offer the same kind of operations and do have the same semantics. For simplicity in this document the functionality of the condition class is described only via ara::diag::Condition. Whenever a functionality of the ara::diag::Condition class is described, the same applies for the operation on the ara::diag::MultipleCondition.

[SWS_DM_01735] MultipleCondition ConditionHandleType Generation

Upstream requirements: RS_Diag_04192

⌈The ConditionHandleType is a numerical value created by the following rules:

- If no DiagnosticMultipleConditionPortMapping.overrideId exists, in alphabetical order of DiagnosticCondition shortname the DM assigns for the first DiagnosticCondition a number of 0x010000 and increments this number for each further DiagnosticCondition.
- If DiagnosticMultipleConditionPortMapping.overrideId exists, the number idOverride shall be used instead.
- If an automatic created number collides with a preconfigured idOverride value, the next free value shall be chosen.

⌋

The idOverride is used to allow the application developer or integrator to use/reuse a fixed number from the application that is not changed if the auto generated number changes due to an update of the input configuration with new events or changed names. The numbers are greater or equal to 0x010000, reserving the range between 0x0000 and 0xFFFF for future extensions and definitions.

[SWS_DM_01736] Invalid ConditionHandleType

Upstream requirements: RS_Diag_04192

⌈If any ara::diag::MultipleCondition method with input parameter conditionHandle is called and this instance of MultipleCondition has not mapped DiagnosticMultipleConditionPortMapping that would create the provided ConditionHandleType, the DM shall return kInvalidArgument.⌋

7.3.4.1.5 Debouncing

Debouncing of reported events is the capability of the DM to filter out undesirable noise reported by monitors. This is used to mature the result of the monitor.

Debouncing means that a report from a monitor does not immediately lead to a change of the Event status bit "kTestFailed" but that a delaying threshold value must be reached before. This results in the states for ara::diag::DebouncingState. If this threshold value is reached (FDC-equivalent is +127 (FDC_{max}) or -128 (FDC_{min})), the ara::diag::DebouncingState is either kFinallyDefective or kFinallyHealed. This finally also leads than to a change of the Event status bit "kTestFailed".

There are two kinds of different debounce algorithms implemented by the DM:

- Counter-based debouncing (see 7.3.4.1.5.1 “Counter-based debouncing”))
- Time-based debouncing (see 7.3.4.1.5.2 “Time-based debouncing”))

Besides the here described debouncing algorithms within the DM implementation, there is also the possibility to do the debouncing monitor-internal within the AA (compare [SWS_DM_00548]). This functionality is not part of the DM, and the use case in described in 7.3.4.1.5.3 “Monitor-internal debouncing”).

If the FDC for an event with monitor-internal debouncing is needed, the registered callback function getFaultDetectionCounter from the constructor ara::diag::Monitor::Monitor is used.

Which algorithm is used can be configured on a per event basis.

The DM is not supporting debouncing for an event:

- If an event is not referenced by any DiagnosticEventToDebounceAlgorithmMapping.diagnosticEvent
- If an event is referenced by DiagnosticEventToDebounceAlgorithmMapping and DiagnosticDebounceAlgorithmProps.debounceAlgorithm is configured to DiagEventDebounceMonitorInternal.

Note: [constr_10418] enforces the existence of attribute DiagnosticDebounceAlgorithmProps.debounceAlgorithm.

A monitoring application will call the method ara::diag::Monitor::ReportMonitorAction with the parameter action set to kPrepassed or kPrefailed for events, that are debounced by the DM.

For debouncing configuration values associated with diagnostic events, two sources can be referenced - 1) DEXT, 2) Monitor constructor. The debouncing configuration provided to the Monitor constructor is considered as default and is overwritten if a debouncing configuration is provided in DEXT. As a consequence, any debouncing configuration in DEXT needs to have a meaningful value.

[SWS_DM_01099] Debouncing parameters from DEXT

Upstream requirements: RS_Diag_04068

⌈If Debouncing parameters are present in DEXT and Debouncing Algorithm is fitting (i.e. Timer Based or Counter based), DM shall refer to the DEXT values as given in the requirements below and ignore any values derived from Monitor Constructors.⌋

[SWS_DM_01101] Debouncing parameters from Monitor Constructor

Upstream requirements: RS_Diag_04068

⌈If Debouncing parameters and methods are not present in DEXT, DM shall use the Monitor Constructor values and Algorithm, as provided. In these instances, debouncing relevant DEXT parameters in the following requirements shall be replaced by the Monitor Constructor values.⌋

7.3.4.1.5.1 Counter-based debouncing

Counter-based debouncing is done on a per event based counting policy of reported kPrepassed or kPrefailed from diagnostic monitors. Per event an internal debounce counter is used. Passed or failed event states for events are calculated by evaluating configured thresholds of the internal debounce counter.

[SWS_DM_00014] Use of counter-based debouncing for events

Upstream requirements: RS_Diag_04068

⌈A DiagnosticEvent shall be subject to counter-based debouncing if the DiagnosticEvent is referenced in the role diagnosticEvent by a DiagnosticEventToDebounceAlgorithmMapping, where the referenced debounceAlgorithm aggregates a DiagEventDebounceCounterBased in the role debounceAlgorithm.⌋

[SWS_DM_00018] Internal debounce counter init

Upstream requirements: RS_Diag_04068

⌈Upon start-up, the DM shall initialize the event's internal debounce counter to '0'.⌋

[SWS_DM_00017] Calculation of the FDC based on the internal debounce counter

Upstream requirements: RS_Diag_04125, RS_Diag_04190

⌈The DM shall calculate the FDC according to UDS, based on the value and range of the internal debounce counter by linear mapping in order to achieve a value range of -128 ... +127.⌋

[SWS_DM_00875] Internal debounce counter incrementation

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈The DM shall increment the event's internal debounce counter by the configured step-size value of DiagEventDebounceCounterBased.counterIncrementStepSize, when the related monitor calls the method ara::diag::Monitor::ReportMonitorAction with the parameter action set to kPrefailed. If after the incrementation the debounce counter value is greater than the threshold DiagEventDebounceCounterBased.counterFailedThreshold, the DM shall set the debounce counter value equal to the threshold.⌋

[SWS_DM_00024] Qualified failed event using counter-based debouncing

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈If the internal debounce counter is greater or equal to DiagEventDebounceCounterBased.counterFailedThreshold the DM shall process the event as kFinallyDefective.⌋

[SWS_DM_00876] Internal debounce counter decrementation

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈The DM shall decrement the event's internal debounce counter by the configured step-size value of DiagEventDebounceCounterBased.counterDecrementStepSize, when the related monitor calls the method ara::diag::Monitor::ReportMonitorAction with the parameter action set to kPrepassed. If after the decrementation the debounce counter value is lower than the threshold DiagEventDebounceCounterBased.counterPassedThreshold, the DM shall set the debounce counter value equal to the threshold.⌋

[SWS_DM_00025] Qualified passed event using counter-based debouncing

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈If the internal debounce counter is less or equal to DiagEventDebounceCounterBased.counterPassedThreshold the DM shall process the event as kFinallyHealed.⌋

[SWS_DM_00021] Direct failed qualification of counter-based events

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈If the monitor reports kFailed, the DM shall set the internal debounce counter to the value DiagEventDebounceCounterBased.counterFailedThreshold and process the event as kFinallyDefective.⌋

[SWS_DM_00029] Direct passed qualification of counter-based events

Upstream requirements: RS_Diag_04125, RS_Diag_04068

⌈If the monitor reports kPassed, the DM shall set the internal debounce counter to the value DiagEventDebounceCounterBased.counterPassedThreshold and process the event as kFinallyHealed.⌋

[SWS_DM_00022] Internal debounce counter jump up behavior

Upstream requirements: RS_Diag_04068

⌈If DiagEventDebounceCounterBased.counterJumpUp is set to true for an event, the DM shall set the event's internal debounce counter to DiagEventDebounceCounterBased.counterJumpUpValue if kPrefailed is reported for this event and the current internal debounce counter value is less than DiagEventDebounceCounterBased.counterJumpUpValue. After setting the internal debounce counter to DiagEventDebounceCounterBased.counterJumpUpValue the processing according to [SWS_DM_00875] shall be done.⌋

[SWS_DM_00023] Internal debounce counter jump down behavior

Upstream requirements: RS_Diag_04068

⌈If kPrepassed is reported for an event and the current internal debounce counter value is greater than DiagEventDebounceCounterBased.counterJumpDownValue and counterJumpDown is set to true for this event, the DM shall set the event's internal debounce counter to DiagEventDebounceCounterBased.counterJumpDownValue. After setting the internal debounce counter to DiagEventDebounceCounterBased.counterJumpDownValue the processing according to [SWS_DM_00876] shall be done.⌋

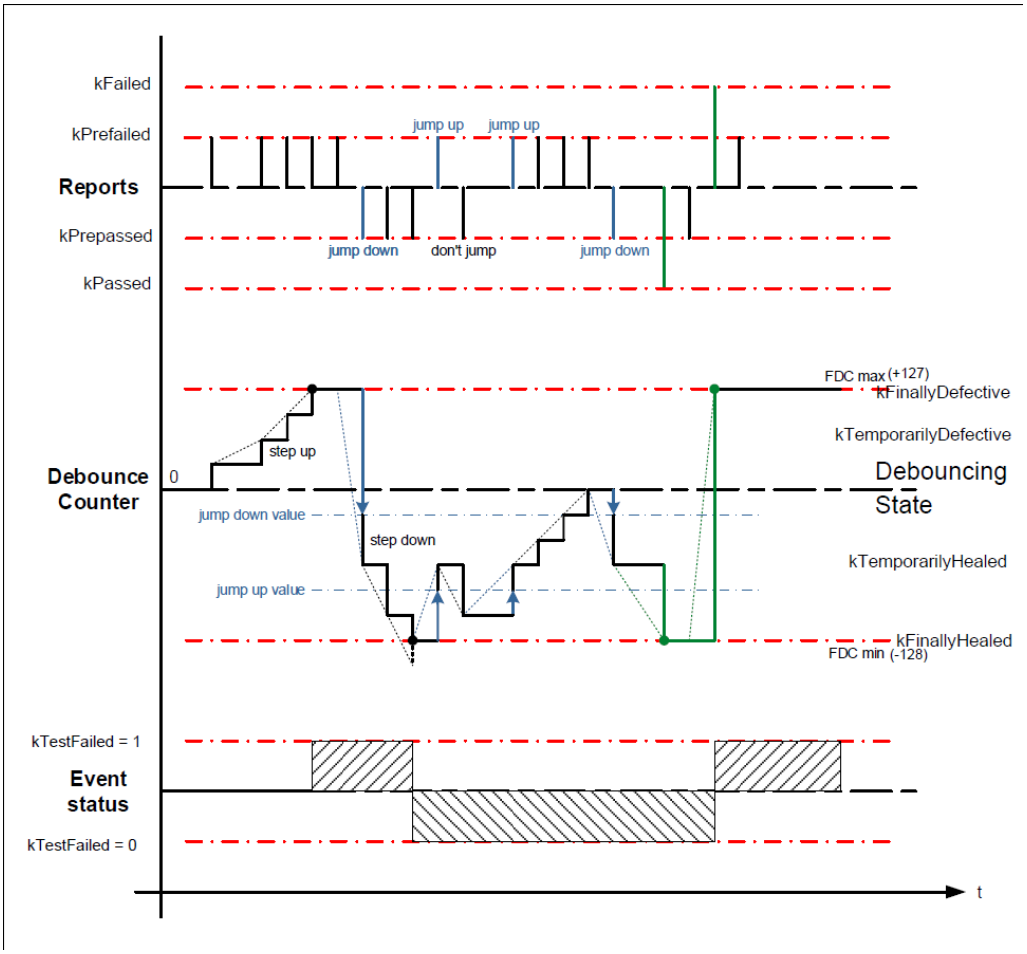

Figure 7.6: Counter-based debouncing

7.3.4.1.5.2 Time-based debouncing

Time-based debouncing is done on a per event based counting policy of reported kPrepassed or kPrefailed from diagnostic monitors. Per event an internal debounce timer value is used. The Event status bit "kTestFailed" for events are calculated by evaluating configured thresholds of the internal debounce timers.

[SWS_DM_00015] Use of timer based debouncing for events

Upstream requirements: RS_Diag_04225

⌈The existence of a DiagnosticEventToDebounceAlgorithmMapping with an aggregation of DiagEventDebounceTimeBased by the referenced DiagnosticDebounceAlgorithmProps.debounceAlgorithm shall activate a time-based debouncing for this event.⌋

[SWS_DM_00085] Internal debounce timer init

Upstream requirements: RS_Diag_04225

⌈The DM shall initialize the event's internal debounce timer to '0' upon start-up.⌋

Note: debounceCounterStorage is not supported for time-based debouncing

[SWS_DM_00030] Calculation of the FDC based on the internal debounce timer

Upstream requirements: RS_Diag_04225, RS_Diag_04190

⌈The DM shall calculate the FDC according to ISO 14229-1, based on the value and range of the internal debounce timer by linear mapping in order to achieve a value range of -128 ... +127.⌋

The debounce timer is used to run upon a kPrefailed towards the qualified failed and upon a kPrepassed towards a qualified passed.

[SWS_DM_00877] Starting time-based event debouncing towards kFinallyDefective

Upstream requirements: RS_Diag_04225

⌈The DM module shall start the debounce timer when the related monitor calls the method ara::diag::Monitor::ReportMonitorAction with the parameter action set to kPrefailed to qualify the reported event as kFinallyDefective after DiagEventDebounceTimeBased.timeFailedThreshold only when the following conditions are met:

- The debounce timer for the event is not already running towards kFinallyDefective.
- The event is not already qualified as kFinallyDefective.

⌋

[SWS_DM_00033] Debounce timer behavior upon reported `kFailed`

Upstream requirements: RS_Diag_04225

⌈If the `monitor` reports `kFailed`, the `DM` shall set the debounce timer value to `DiagEventDebounceTimeBased.timeFailedThreshold` and process the event as `kFinallyDefective`.⌋

[SWS_DM_00878] Starting time-based event debouncing towards `kFinallyHealed`

Upstream requirements: RS_Diag_04225

⌈The `DM` module shall start the debounce timer when the related `monitor` calls the method `ara::diag::Monitor::ReportMonitorAction` with the parameter `action` set to `kPrepassed` to qualify the reported event as `kFinallyHealed` after `DiagEventDebounceTimeBased.timePassedThreshold` only when the following conditions are met:

- The debounce timer for the event is not already running towards `kFinallyHealed`.
- The event is not already qualified as `kFinallyHealed`.

⌋

[SWS_DM_00036] Debounce timer behavior upon reported `kPassed`

Upstream requirements: RS_Diag_04225

⌈If the `monitor` reports `kPassed`, the `DM` shall set the debounce timer value to `DiagEventDebounceTimeBased.timePassedThreshold` and process the event as `kFinallyHealed`.⌋

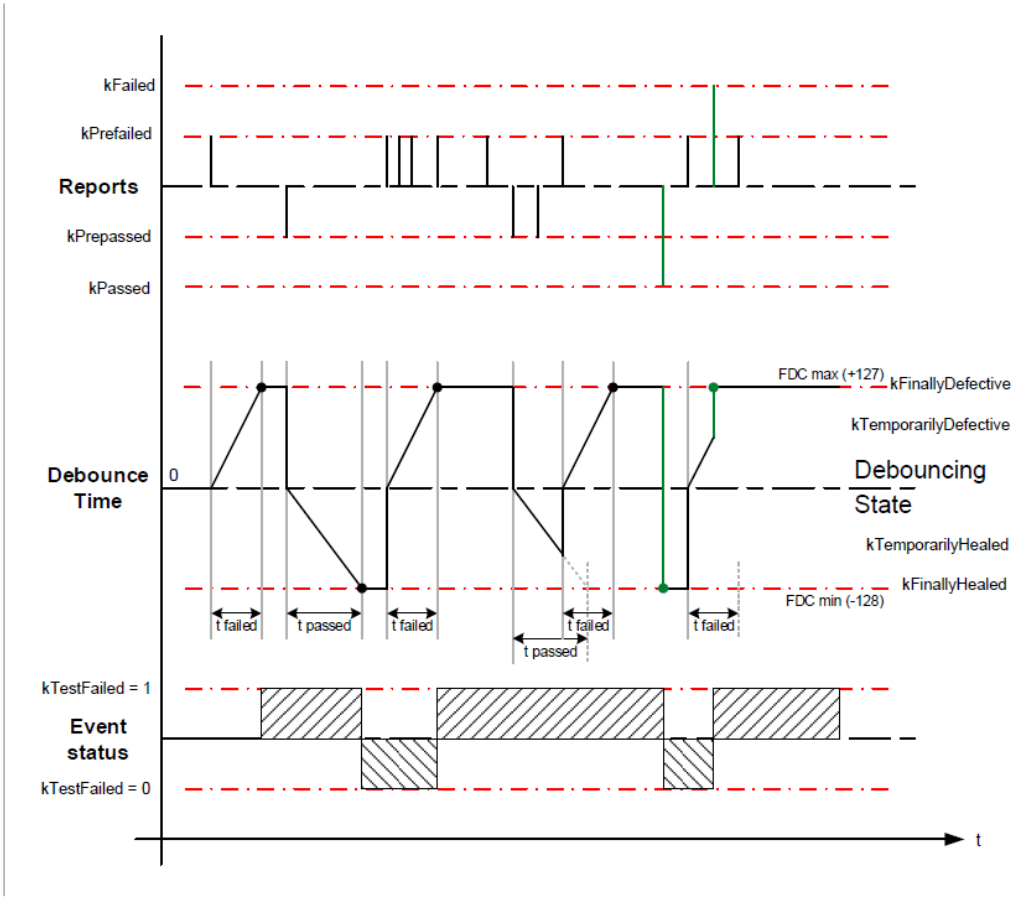

Figure 7.7: Timer based debouncing

[SWS_DM_00880] Debounce time freeze request

Upstream requirements: RS_Diag_04068, RS_Diag_04225

⌈If the `ara::diag::Monitor::ReportMonitorAction` method of a `ara::diag::Monitor` instance is called with the parameter `action` set to `kFreezeDebouncing`, for events with `DiagEventDebounceTimeBased` debouncing, the `DM` shall freeze the related debounce timer of the corresponding `event`.⌋

Freezing of the timer is only supported for events with `DiagEventDebounceTimeBased` debouncing.

[SWS_DM_00038] Continuing a frozen debounce timer

Upstream requirements: RS_Diag_04225

⌈If a debounce timer is frozen (i.e. the corresponding `monitor` has called `ara::diag::Monitor::ReportMonitorAction` with the parameter `action` set to `kFreezeDebouncing`) and a new `kPrepassed` or `kPrefailed` is reported for this event, the `DM` module shall continue running the debounce timer starting with the frozen value.⌋

7.3.4.1.5.3 Monitor-internal debouncing

Monitor-internal debouncing is completely left to the diagnostic monitor `AA` which just reports the final debouncing results by calling `ara::diag::Monitor::ReportMonitorAction` on a per event base. Consequently, there is no DM-internal debouncing logic for these events. Monitor-internal events are modeled in `DEXT` by `DiagnosticEvents` referenced in the role `diagnosticEvent` by a `DiagnosticEventToDebounceAlgorithmMapping`, where the referenced `debounceAlgorithm` aggregates a `DiagEventDebounceMonitorInternal` in the role `debounceAlgorithm`. The `DM` processes these events according to [SWS_DM_01025] and the below requirement(s) apply additionally.

There are monitors that do not debounce at all and directly provide a qualified monitor result to the `DM`. In that case there is no `FDC` in the system and the `DM` can derive the `FDC` value directly from the reported monitor result.

[SWS_DM_01971] `FDC` value for monitors without debouncing ⌈If a monitor was initialised by `ara::diag::Monitor::Monitor` or , `ara::diag::MultipleMonitor::ConfigureMonitor` the `DM` shall derive the `FDC` value by the following rules:

- -128 in case TestFailed=0 and TestNotCompletedThisCycle = 0
- 0 in case TestNotCompletedThisCycle = 1
- 127 in case TestFailed=1 and TestNotCompletedThisCycle = 0."

⌋

[SWS_DM_01267] Reporting `kFdcThresholdReached` for monitor internal debouncing

Upstream requirements: RS_Diag_04127

⌈If `ara::diag::Monitor::ReportMonitorAction` is called with the parameter action set to `kFdcThresholdReached`, the `DM` shall allocate an `event memory` entry and trigger or update the storage of `snapshot records` and/or `extended data records`, if the corresponding `DiagnosticExtendedDataRecord.trigger` and/or `DiagnosticFreezeFrame.trigger` attribute is/are set to `fdcThreshold`, as specified in [SWS_DM_01085], [SWS_DM_01086] and [SWS_DM_00895].⌋

[SWS_DM_01268] Value of FaultDetectionCounter in case of monitor internal debouncing

Upstream requirements: RS_Diag_04068

⌈If monitor internal debouncing is used, The DM shall use the value returned after calling the registered callback function `getFaultDetectionCounter` of the `ara::diag::Monitor::Monitor` constructor as `FDC`.⌋

7.3.4.1.5.4 Debounce algorithm reset

In some situations the application might want to reset the debouncing or to freeze it. The `DM` provides the parameter `action` with value `kResetDebouncing` or `kFreezeDebouncing` for the method `ara::diag::Monitor::ReportMonitorAction` of class `ara::diag::Monitor` to provide some means of external control of the internal debounce counter.

[SWS_DM_01583] Resetting counter-based debouncing

Upstream requirements: RS_Diag_04068, RS_Diag_04225

⌈If a monitor is configured with `DiagEventDebounceCounterBased` and ReportMonitorAction is called with `kResetDebouncing`, the `DM` shall reset the debounce counter and `FDC` to zero.⌋

[SWS_DM_01584] Resetting time-based debouncing

Upstream requirements: RS_Diag_04068, RS_Diag_04225

⌈If a monitor is configured with `DiagEventDebounceTimeBased` and ReportMonitorAction is called with `kResetDebouncing`, the `DM` shall stop the timer and reset `FDC` to zero.⌋

[SWS_DM_00039] Resetting the internal debounce counter upon restarting an operation cycle

Upstream requirements: RS_Diag_04068, RS_Diag_04225

⌈If an operation cycle is restarted, the DM shall reset the internal debounce counter for all events referenced by DiagnosticEventToOperationCycleMapping.diagnosticEvent and referencing the restarted operation cycle by DiagnosticEventToOperationCycleMapping.operationCycle.⌋

[SWS_DM_00086] Resetting the internal debounce counter after clearing DTC

Upstream requirements: RS_Diag_04068, RS_Diag_04225

⌈If the DM executes a ClearDTC command, the DM shall reset the internal debounce counter for all events that have a DiagnosticEventToTroubleCodeUdsMapping to one of the cleared DTCs.⌋

7.3.4.1.5.5 Dependencies to enable conditions

As described in section 7.3.4.1.4 enable conditions are used to suppress the result of reported event status information. Enable Conditions have also effect on the debouncing behavior of the DM.

[SWS_DM_00882] Enable condition influence on debouncing behavior

Upstream requirements: RS_Diag_04192, RS_Diag_04125

⌈If the enable condition state for an event is changed to not fulfilled as defined by [SWS_DM_00568] and a DM internal timer or counter based debouncing is used, the DM shall reset the according internal debounce counter for this event.⌋

7.3.4.1.5.6 Dependencies to UDS service 0x85 ControlDTCSettings

[SWS_DM_00378] ControlDTCSetting influence

Upstream requirements: RS_Diag_04159, RS_Diag_04125

⌈If ControlDTCSetting is set to disabled according to [SWS_DM_00910] and a DM internal timer or counter based debouncing is used for an event, the DM shall reset the event's internal debounce counter and freeze it for the time the ControlDTCSetting is set to disabled.⌋

7.3.4.1.6 Event Status processing

The 'Event Status processing' is the DMs ability to record and retain Events, Event status and associated data.

The DM provides means to other SW parts in order to control the Event status bits and is therefore the first processing step after the monitors reporting.

[SWS_DM_01024] Event Status processing

Upstream requirements: RS_Diag_04151

⌈The DM shall process the existing Event status bits ([SWS_DM_00643]) like the processing of the corresponding UDS DTC status bits as specified by the ISO 14229-1 [1] standard.⌋

ISO 14229-1 Annex D generally defines UDS DTC status byte handling and the corresponding triggerings for them. The three corresponding Event status bits are handled in the same way. The following requirements map interfaces and configuration parameters of the DM to generic Event status bit transition descriptions.

[SWS_DM_01025] Event status bit transitions triggered by test results

Upstream requirements: RS_Diag_04151

⌈The DM shall process the Event status bits triggered by the test results (kPassed or kFailed) reported via the function ara::diag::Monitor::ReportMonitorAction of the corresponding ara::diag::Monitor instance. Here, kPassed shall be used as "*TestResult [Passed]*" and kFailed as "*TestResult [Failed]*" as described in [ISO 14229-1] Annex D.2.⌋

Note that if debouncing for an event is configured, kPrepassed or kPrefailed reported via ara::diag::Monitor::ReportMonitorAction, trigger debounce mechanisms (see section 7.3.4.1.5). These status reports do not have direct impact on the Event status bits. If the status of an event gets fully qualified after debouncing (i.e. kFinallyHealed or kFinallyDefective), this information has the same impact on the Event status bits as if kPassed or kFailed would have been reported via ara::diag::Monitor::ReportMonitorAction.

[SWS_DM_01026] Resetting the status of an Event

Upstream requirements: RS_Diag_04151

⌈If the parameter action in the function ara::diag::Monitor::ReportMonitorAction is set to kResetTestFailed, the DM shall update the Event status by setting **only** the "kTestFailed" bit to FALSE and leave all other bits unchanged.⌋

Rationale: This is an AUTOSAR-specific additional reset condition for the 'kTestFailed' bit of the Event status bits.

[SWS_DM_01027] Event status bit transitions triggered by operation cycle restarting

Upstream requirements: RS_Diag_04178, RS_Diag_04182

⌈If the function ara::diag::OperationCycle::RestartOperationCycle is called, the DM shall reset the Event status bits:

- kTestFailedThisOperationCycle = 0;
- kTestNotCompletedThisOperationCycle = 1;

for all events having a DiagnosticEventToOperationCycleMapping to the restarted operation cycle.⌋

[SWS_DM_01028] Event status bit transitions triggered by ClearDiagnosticInformation UDS service

Upstream requirements: RS_Diag_04180

⌈If the clearing of a DTC is triggered by the UDS service 0x14 *ClearDiagnosticInformation*, the DM shall reset the Event status bits to:

- kTestFailed = 0;
- kTestFailedThisOperationCycle = 0;
- kTestNotCompletedThisOperationCycle = 1;

⌋

7.3.4.1.7 Event status change notifications

[SWS_DM_00886] Observability of the Event status byte

Upstream requirements: RS_Diag_04183

⌈If an AA calls the function ara::diag::Event::GetEventStatus, the DM shall provide the current status of this event from the corresponding ara::diag::Event instance.⌋

[SWS_DM_01029] Notification about Event status changes

Upstream requirements: RS_Diag_04183

⌈If the AA has registered for a Event status change notification via the function ara::diag::Event::SetEventStatusChangedNotifier of the corresponding ara::diag::Event instance, the DM shall call this notifier for each status change of this Event.⌋

7.3.4.1.8 Event occurrence

Event occurrence is defined as the number of repetitions of the same error. An occurrence counter exists per event memory entry as part of the event related datas to that entry. The event occurrence counter is expected to have a size of one byte.

[SWS_DM_00945] Occurrence Counter initial value

Upstream requirements: RS_Diag_04067

⌈If a new `event memory` entry is created, the `DM` module shall initialize the associated occurrence counter with the value '1'.⌋

[SWS_DM_00946] Occurrence Counter increment strategy 'testFailed'-only

Upstream requirements: RS_Diag_04067

⌈If the configuration parameter `DiagnosticCommonProps.occurrenceCounterProcessing` is set to `testFailedBit` and a certain `event` is already stored in the `event memory`, for each transition of `UDS DTC status bit` "`kTestFailed`" from '0' to '1', the `DM` module shall trigger the increment of the associated occurrence counter by one, regardless of `UDS DTC status bit` "`kConfirmedDTC`" state.⌋

[SWS_DM_00947] Occurrence Counter increment strategy 'confirmedDtcBit'

Upstream requirements: RS_Diag_04105

⌈If the configuration parameter `DiagnosticCommonProps.occurrenceCounterProcessing` is set to `confirmedDtcBit` and a certain `event` is already stored in the `event memory` and the `UDS DTC status bit` "`kConfirmedDTC`" is equal to '1', for each transition of `UDS DTC status bit` "`kTestFailed`" from '0' to '1', the `DM` module shall trigger the increment of the associated occurrence counter by one.⌋

[SWS_DM_00948] Occurrence Counter upper limit

Upstream requirements: RS_Diag_04125

⌈If an occurrence counter has reached its maximum value of 255 and the conditions to increment this occurrence counter are met, the `DM` module shall latch this occurrence counter at its maximum value.⌋

A rollover of the occurrence counter byte from 255 to 0 must be avoided.

[SWS_DM_01349] Consecutive registration of notifier with SetEventStatusChangedNotifier()

Upstream requirements: RS_Diag_04125

⌈In case of a consecutive call of `ara::diag::Event::SetEventStatusChangedNotifier` of the corresponding `ara::diag::Event` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.4.2 Condition Mangement

The DM supports the use of conditions in situations where the processing of event or clearing DTCs is not desired. For that purpose the ara::diag::Condition is used. Examples for the use of conditions are:

- EnableConditions (see 7.3.4.1.4 “EnableConditions”)
- ClearDTCConditions (see 7.3.4.4.5.2 “ClearConditions”)

At the time the application calls the ara::diag::Condition::SetCondition method, the connection between the interface and the DM might not be active. Typical situations are startup or communication interruptions. To ensure that no crucial data is lost, the conditions interface is caching the result and bridges the time-span, until the connection to the DM is up and running again.

[SWS_DM_01093] Caching of conditions

Upstream requirements: RS_Diag_04192

⌈If the function SetCondition() is called and the DM is currently not ready to process the reported condition state, the DM shall cache the latest reported condition and evaluate it when the connection to the DM is (re-)established.⌋

7.3.4.3 Operation Cycles Management

The DM supports operation cycles according to ISO 14229-1[1]. Operation cycles have direct effect on the UDS DTC status byte and the event memory behavior.

Examples of typical operation cycles are:

- Ignition on/off cycles
- Power up/power down cycle
- Accumulated operating time cycles
- ...

Operation cycles are managed by the AA, the DM is notified about the restart of an operation cycle by using the API interface function ara::diag::OperationCycle::RestartOperationCycle.

[SWS_DM_01103] Caching of RestartOperationCycle

Upstream requirements: RS_Diag_04178, RS_Diag_04182

⌈If the function `ara::diag::OperationCycle::RestartOperationCycle` is called and the `DM` is currently not ready to process the reported restart of the `operation cycle`, the `DM` shall cache one reported `operation cycle` restart and evaluate it when the connection to the `DM` is (re-)established.⌋

[SWS_DM_01104] Operation Cycle restart

Upstream requirements: RS_Diag_04178

⌈The `DM` shall only restart `operation cycles` after the previous restart has been fully processed. In other words this means that there is no `operation cycle` restart queue.⌋

[SWS_DM_01105] Restart OperationCycle during the processing of previous call

Upstream requirements: RS_Diag_04178

⌈If the `DM` is still processing the previous `ara::diag::OperationCycle::RestartOperationCycle` call of the same `operation cycle`, the `DM` shall ignore the current method call.⌋

[SWS_DM_01358] Consecutive registration of notifier with OperationCycle::SetNotifier()

Upstream requirements: RS_Diag_04178, RS_Diag_04186

⌈In case of a consecutive call of `ara::diag::OperationCycle::SetNotifier` of the corresponding `ara::diag::OperationCycle` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.4.4 Event memory

The `event memory` is the database for faults detected by the system. It stores status information for `events`, `DTCs` and `DTC` related data.

The term "`event memory`", wherever used in this specification, refers to the term "`fault memory`" as specified in ISO 14229-1 [1]. The DM "`event memory`" is compliant to the "`fault memory`" in ISO.

There can be multiple `event memories` handled by the `DM`.

[SWS_DM_00055] Supported event memories

Upstream requirements: RS_Diag_04214, RS_Diag_04150, RS_Diag_04058

⌈The `DM` shall support the

- primary event memory
- up to 256 user-defined event memories

according to ISO 14229-1[1].⌋

Note: From DM perspective the primary and user defined fault memories are all instances of a database. Each instance is treated in exactly the same way.

[SWS_DM_01960] Identical fault memory behavior

Upstream requirements: RS_Diag_04131

⌈The user defined memory shall have the same behavior as the primary memory (event prioritization, aging, displacement).⌋

[SWS_DM_00911] Instances of DTCInformation interface

Upstream requirements: RS_Diag_04214, RS_Diag_04150

⌈The DM shall link each instance of the ara::diag::DTCInformation class with the mapped DiagnosticMemoryDestination referenced by the corresponding DiagnosticMemoryDestinationPortMapping.⌋

[SWS_DM_00056] Availability of the primary event memory

Upstream requirements: RS_Diag_04150, RS_Diag_04058

⌈The DM shall support the primary event memory if a DTC exists having a DiagnosticMemoryDestinationPrimary referenced by its DiagnosticTroubleCodeProps.diagnosticMemory.⌋

[SWS_DM_00057] Availability of a user-defined event memory

Upstream requirements: RS_Diag_04214, RS_Diag_04058

⌈The DM shall support the user-defined event memory with the number DiagnosticMemoryDestinationUserDefined.memoryId if a DTC exists having a DiagnosticMemoryDestinationUserDefined with that user-defined number referenced by its DiagnosticTroubleCodeProps.diagnosticMemory.⌋

The size of the different event memories is configurable by the DM configuration parameters.

[SWS_DM_00920] Configuration of the event memory size

Upstream requirements: RS_Diag_04064

⌈The DM shall provide event memories of a size according to the configuration parameter DiagnosticMemoryDestination.maxNumberOfEventEntries, where each single event memory entry is to be understood as the complete data set that

belongs to a DTC, including counters, cycles, snapshot records and extended data records.⌋

If there are limitations to the event memory size, an overflow can occur as a consequence. Therefore the DM provides an overflow indication in case the event memory overflow occurs and a displacement strategy.

7.3.4.4.1 DTC Introduction

A diagnostic trouble code (DTC) defines a unique identifier mapped to a diagnostic event via DiagnosticEventToTroubleCodeUdsMapping. The DTC is used by diagnostics, including e.g. UDS communication with an external tester, to uniquely identify data within the event memory database.

[SWS_DM_00060] Set of supported DTCs

Upstream requirements: RS_Diag_04201

⌈The existence of a DiagnosticTroubleCodeUds indicates that the DM shall support this DTC.⌋

Note: Due to DM restrictions the 'DiagnosticTroubleCodeObd' and 'DiagnosticTroubleCodeJ1939' are not supported.

7.3.4.4.1.1 Format

The DTC itself is a 3 byte value, that could have different interpretations.

[SWS_DM_00058] DTC interpretation format

Upstream requirements: RS_Diag_04157

⌈The DM shall use one internal DTC format interpretation that is defined in DiagnosticMemoryDestinationPrimary.typeOfDtcSupported.⌋

Note: Refers to [TPS_DEXT_01008] in [3].

[SWS_DM_CONSTR_00059] Restriction on supported DTC format

Upstream requirements: RS_Diag_04201

⌈The DM shall support the following literals from interpreted DiagnosticMemoryDestinationPrimary.typeOfDtcSupported (see also [SWS_DM_00058])

- iso11992_4
- iso14229_1

- saeJ2012_da

Further information about the format mapping is defined in [SWS_DM_00062].

The following literals are **not** supported by the DM:

- iso15031_6
- saeJ1939_73

⌋

7.3.4.4.1.2 Groups

Besides the term DTC, diagnostics uses DTC groups to address a range of single DTCs. A DTC group is defined by using a dedicated DTC value out of the range of valid DTCs to identify the group of DTCs.

A definition of valid DTC groups is provided by ISO 14229-1 [1] - Annex D.1. The DTC group is used in diagnostic just as any other DTC value, the DM internally resolves the DTC group and applies the requested operation to all DTCs of that group. The most common DTC group is the group of all DTCs, assigned to the DTC value 0xFFFFFF.

[SWS_DM_00064] Definition of DTC groups

Upstream requirements: RS_Diag_04117, RS_Diag_04115

⌈The existence of a DiagnosticTroubleCodeGroup shall define the existence of the DTC group with the DTC identifier DiagnosticTroubleCodeGroup.groupNumber⌋

Note: Refers to [TPS_DEXT_03014] in [3].

[SWS_DM_00065] Always supported availability of the group of all DTCs

Upstream requirements: RS_Diag_04117

⌈The DM shall provide by default the DTC group 'GroupOfAllDTCs' assigned to the DTC group identifier 0xFFFFFF. This DTC group contains always all configured DTCs.⌋

[SWS_DM_CONSTR_00082] Restriction on the configuration of the DTC group GroupOfAllDTCs

Upstream requirements: RS_Diag_04117

⌈The DM shall ignore any configuration of a DiagnosticTroubleCodeGroup.groupNumber with a value of 0xFFFFFF.⌋

A configuration of the DTC group 0xFFFFFF via DiagnosticTroubleCodeGroup.groupNumber is not required. Within the DM basically all services and diagnostic requests having a DTC as input parameter accept also DTC group. As result of this, the operation is applied on all DTCs of that DTC group. To provide the reader a clear understanding if the DTC also can be a DTC group, it is explicitly mentioned in this specification. In case a DTC group is also valid, the DTC group definition of this chapter applies.

7.3.4.4.1.3 Priority

DTC priority is defined as a ranking based upon the level of importance. It is used to determine which entry may be removed from the event memory in case the number of already stored events exceeds the maximum number of memory entries (event memory is full) (7.3.4.4.10).

It is also used for internal DiagnosticDataElements connected to a DTCs extended data record.

Each supported DTC has a priority assigned to it: DiagnosticTroubleCodeProps.priority;

[SWS_DM_00916] Priority values

Upstream requirements: RS_Diag_04118, RS_Diag_04071

⌈If a DTC has a priority value of 1 and displacement needs to be applied, the DM shall consider this as the highest priority. A higher value shall define a lower priority.⌋

[SWS_DM_CONSTR_00961] Limits of priority values

Upstream requirements: RS_Diag_04118, RS_Diag_04071

⌈The DM shall only support DiagnosticTroubleCodeProps.priority values in the range of 1..255.⌋

7.3.4.4.2 UDS DTC Status

7.3.4.4.2.1 Status processing

The 'UDS DTC Status processing' is the DMs ability to record and retain UDS status and associated interactions with other SW parts.

Thus the 'UDS Status processing' is an essential part of the DM functionality and is the second processing step after the Event status handling, as defined in 7.3.4.1.6. The DM provides means to other SW parts in order to control the UDS DTC status bits.

[SWS_DM_00213] DTC status processing

Upstream requirements: RS_Diag_04067

⌈The DM shall process the UDS DTC status byte harmonizing with the ISO 14229-1[1] standard.⌋

ISO 14229-1 Annex D generally defines UDS DTC status byte handling and the corresponding triggerings for them. The following requirements map interfaces and configuration parameters of the DM to generic UDS DTC status bit transition descriptions.

[SWS_DM_00883] UDS DTC status bit transitions triggered by test results

Upstream requirements: RS_Diag_04067

⌈The DM shall process the UDS DTC status byte based on the Event status bits.⌋

[SWS_DM_00217] UDS DTC status bit transitions triggered by ClearDiagnosticInformation UDS service

Upstream requirements: RS_Diag_04180, RS_Diag_04067

⌈If the clearing of a DTC is triggered by the UDS service 0x14 *ClearDiagnosticInformation*, the DM shall process the UDS DTC status byte according to ISO 14229-1[1].⌋

[SWS_DM_00218] kConfirmedDTC (Bit3) calculation

Upstream requirements: RS_Diag_04136, RS_Diag_04067, RS_Diag_04151

⌈If the kTestFailed bit (Bit0) of the UDS DTC status bit has a transition from 0 to 1, the DM shall set the kConfirmedDTC bit (Bit3), in case a DiagnosticEvent.confirmationThreshold is configured and the trip counter is equal to confirmationThreshold-1.⌋

The trip counter is processed according to ISO 14229-1[1], Annex D2. The trip counter is always increased by one at the end of any operation cycle with kTestFailedThisOperationCycle (Bit1) set to 1. Or in other words, a confirmationThreshold value of 1 or no configured value confirms the DTC at the same time, the test failed bit is set to 1.

If Aging is supported for a DTC, the status is handled according to [SWS_DM_00243].

If there is an indicator mapped to the DTC, the "kWarningIndicatorRequested" bit is handled as described in section 7.3.4.4.2.3.

[SWS_DM_01037] Behavior of not configured DiagnosticEvent.confirmationThreshold

Upstream requirements: RS_Diag_04067

⌈If the optional parameter DiagnosticEvent.confirmationThreshold is not configured, the DM shall use a default value of '1' for that parameter.⌋

This means that a confirmedDTC is set to '1' along with a reported kFailed monitor action result.

7.3.4.4.2.2 UDS DTC Status change notifications

[SWS_DM_01030] Observability of the UDS DTC status byte

Upstream requirements: RS_Diag_04183

⌈If an AA calls the function ara::diag::DTCInformation::GetCurrentStatus(dtc), the DM shall provide the current status of the given dtc ID from within the corresponding ara::diag::DTCInformation instance.⌋

[SWS_DM_01031] Notification about UDS DTC status changes

Upstream requirements: RS_Diag_04183

⌈If the AA has registered for a UDS DTC status change notification via the function ara::diag::DTCInformation::SetDTCStatusChangedNotifier of the corresponding ara::diag::DTCInformation instance, the DM shall call this notifier on any UDS DTC status change for every single DTC mapped to this fault memory.⌋

[SWS_DM_01350] Consecutive registration of notifier with SetDTCStatusChangedNotifier()

Upstream requirements: RS_Diag_04183

⌈In case of a consecutive call of ara::diag::DTCInformation::SetDTCStatusChangedNotifier of the corresponding ara::diag::DTCInformation instance, DM module shall overwrite the previous registered notifier.⌋

7.3.4.4.2.3 Indicators

Indicators can be associated with a particular DiagnosticEvent. Indicators or 'warning outputs' may consist of lamp(s), displayed text information or similar vendor specific expressions. There can be various DiagnosticEvents per indicator and one DiagnosticEvent can have zero, one or more different indicators assigned.

The indicators are activated and deactivated based on the configured failure and healing cycles per Event.

[SWS_DM_00888] Observability of indicator status

Upstream requirements: RS_Diag_04204

⌈The DM shall provide the status of an indicator via the function ara::diag::Indicator::GetIndicatorState of the corresponding ara::diag::Indicator instance.⌋

The status of an indicator is determined by all the event indicator status combined to the DTC for which the indicator state is requested.

[SWS_DM_01974] Indicator reporting kOnDemand

Upstream requirements: RS_Diag_04204

⌈If at least one indicator state of an event assigned to a DTC is in state kOnDemand, the Diagnostic Server instance shall return kOnDemand as return value of ara::diag::Indicator::GetIndicatorState⌋

[SWS_DM_01975] Indicator reporting kOff

Upstream requirements: RS_Diag_04204

⌈If all indicators of an event assigned to a DTC are in state kOff, the Diagnostic Server instance shall return kOff as return value of ara::diag::Indicator::GetIndicatorState⌋

The DM does not evaluate the DiagnosticConnectedIndicator.behavior. Each DiagnosticIndicator has the states Off and OnDemand as defined in [SWS_DM_00740].

[SWS_DM_00223] Handling of 'warningIndicatorRequested' bit

Upstream requirements: RS_Diag_04204

⌈If a DTC status confirmedDTC bit is set from 0 to 1, the DM shall set the "WarningIndicatorRequested" bit to 1 and set the healing counter to 0.⌋

For confirmation check [SWS_DM_00218].

[SWS_DM_01032] Handling of 'WIR' bit without connected indicators

Upstream requirements: RS_Diag_04204

⌈If there exists no DiagnosticConnectedIndicator configuration item for a DiagnosticEvent and therefore no indicators are assigned to an event and the status

for this event gets confirmed, the DM shall always keep the UDS DTC status bit " kWarningIndicatorRequested" at value '0'.⌋

The DM process the indicator healing based on the DiagnosticConnectedIndicator.healingCycleCounterThreshold configuration parameter of the corrsponding indicator assigned to an event via DiagnosticConnectedIndicator.indicator.

[SWS_DM_00224] Indicator healing

Upstream requirements: RS_Diag_04204

⌈If any indicator is configured, the DM shall set at the end of an operation cycle the UDS DTC status bit 7 (WarningIndicatorRequested) to 0, if the following conditions are fulfilled:

- at least one healingCycleCounterThreshold is greater than 0
- all respective indicator healing cycle counters, which count the number of tested and passed healing OperationCycles, have reached their healingCycleCounterThreshold
- WIRbit is not enabled by calling the API SetLatchedWIRStatus()

⌋

[SWS_DM_01582] Healing counter increment

Upstream requirements: RS_Diag_04204

⌈If the healingCycle of the DiagnosticConnectedIndicator is restarted and kTestFailedThisOperationCycle bit is 0, the DM shall increment the healing counter by 1.⌋

[SWS_DM_01266] Warning Indicator Request Activation

Upstream requirements: RS_Diag_04204

⌈The DM shall set the kWarningIndicatorRequested of the DTC status always in the same time when the kConfirmedDTC bit is set to 1.⌋

This means that the DiagnosticConnectedIndicator.indicatorFailureCycleCounterThreshold is not evaluated by the DM.

[SWS_DM_01359] Consecutive registration of notifier with Indicator::SetNotifier()

Upstream requirements: RS_Diag_04204

⌈In case of a consecutive call of ara::diag::Indicator::SetNotifier of the corresponding ara::diag::Indicator instance, DM module shall overwrite the previous registered notifier.⌋

7.3.4.4.2.4 User controlled WarningIndicatorRequest-bit

In some cases (e.g. controlling a failsafe reaction in an application) the WIR-bit (WarningIndicatorRequest-bit) of a corresponding event in DM shall be set/reset by a dedicated "fail-safe AA".

The "failsafe AA" has to ensure a proper status of the WIR-bit (e.g. regarding to ISO-14229-1[2] or manufacture specific requirements).

The failsafe AA shall report the required WIR-status to DM (via the function ara::diag::Event::SetLatchedWIRStatus of the corresponding ara::diag::Event instance) and has to ensure that the current WIR-status of an event (in DM) fits to the current fail-safe-status in the application:

- fail-safe reaction active: WIR-bit shall be set to "1"
- fail-safe reaction not active: WIR-bit shall be set to "0"

The fail-safe AA has to report the status after every change of its fail-safe state.

Therefore the DM provides the function ara::diag::Event::SetLatchedWIRStatus to set or reset the UDS DTC status bit "kWarningIndicatorRequested" of the related DTC.

Each invocation of the function ara::diag::Event::SetLatchedWIRStatus of an ara::diag::Event instance updates the WIR-bit for the corresponding DTC

Due to not storing the Status-Bit 7 (’warningIndicatorRequested’ bit) on Shutdown, the fail-safe AA has to ensure that the ’warningIndicatorRequest’ bit of a DTC fits to the current failsafe status after initialization of the DM.

Setting the WIR-bit of a DTC can be controlled via SetLatchedWIRStatus() OR by the DM internal WIR-bit handling. (OR-Operation).

[SWS_DM_01033] User controlled set of WIR-bit

Upstream requirements: RS_Diag_04204

⌈If the function ara::diag::Event::SetLatchedWIRStatus is called with parameter status = TRUE, the DM shall set the WIR-bit of the corresponding DTC to "1".⌋

[SWS_DM_01034] User controlled reset of WIR-bit

Upstream requirements: RS_Diag_04204

⌈If the function `ara::diag::Event::SetLatchedWIRStatus` is called with parameter `status` = FALSE and the DM internal WIR-bit handling is also not requesting it, the `DM` shall reset the WIR-bit of the corresponding event to "0".⌋

[SWS_DM_01035] User controlled WIR-bit handling and ControlDTCSetting

Upstream requirements: RS_Diag_04204

⌈If ControlDTCSetting is set to disabled according to [SWS_DM_00910] and the function `ara::diag::Event::SetLatchedWIRStatus` is called, the DM shall not change the status of the WIR-bit and the function shall return `kReportIgnored`.⌋

7.3.4.4.3 Destination

Each `DTC` is stored in one of the supported `event memories` according to [SWS_DM_00056] and [SWS_DM_00057].

[SWS_DM_00083] Event memory destination of an `DTC`

Upstream requirements: RS_Diag_04150, RS_Diag_04214

⌈The existence of `DiagnosticTroubleCodeProps.diagnosticMemory` shall assign all DTCs referencing this `DiagnosticTroubleCodeProps` to the `event memory` referenced by `DiagnosticTroubleCodeProps.diagnosticMemory`.⌋

[SWS_DM_CONSTR_00084] Each `DTC` shall be assigned to an event memory destination

Upstream requirements: RS_Diag_04150, RS_Diag_04214

⌈The `DM` shall only support `DTCs` with a configured `DiagnosticTroubleCodeProps.diagnosticMemory`.⌋

7.3.4.4.4 `DTC` related data

The following sections deal with the `DTC` related data, what includes the triggering and location of freeze frames and extended data records to be stored to. Freeze frames consist of a set of `DIDs` and `extended data records` consist of a set of data elements, which shall be stored in configuration dependent situations.

[SWS_DM_00148] Persistent storage of event memory entries

Upstream requirements: RS_Diag_04211, RS_Diag_04105

⌈The DM shall be able to persistently store the status of all DTCs and for maxNumberOfEventEntries per event memory the DTC related data:

- snapshot data if configured (at least one corresponding DiagnosticTroubleCodeProps.freezeFrame reference exists in the configuration)
- extended data if configured (at least one corresponding DiagnosticTroubleCodeProps.extendedDataRecord reference exists in the configuration)

⌋

[SWS_DM_00969] Padding in case of failed data capturing

Upstream requirements: RS_Diag_04205

⌈If during data collection due to [SWS_DM_01276], [SWS_DM_01277], [SWS_DM_01085], [SWS_DM_01086] or [SWS_DM_00895] an external processor has an error of any source, the DM shall fill the missing data with the padding value 0xFF and trigger a Log and Trace LogError() message.⌋

7.3.4.4.4.1 Triggering for data storage

[SWS_DM_00150] Primary trigger for event memory entry storage

Upstream requirements: RS_Diag_04211, RS_Diag_04105

⌈Creating and storing memory entries (incl. collecting DTC-related data) shall be triggered according to the DiagnosticMemoryDestination.memoryEntryStorageTrigger configuration parameter (see [3]).⌋

Note that for updating snapshot record and extended data information record specific configuration options are available. For details check the following sections.

[SWS_DM_01579] Behavior of not configured DiagnosticMemoryDestination.memoryEntryStorageTrigger

Upstream requirements: RS_Diag_04211, RS_Diag_04105

⌈If the optional parameter DiagnosticMemoryDestination.memoryEntryStorageTrigger is not configured, the DM shall use a default value of 'testFailed' for that parameter.⌋

7.3.4.4.4.2 Storage of snapshot record data

[SWS_DM_00151] snapshot record numeration

Upstream requirements: RS_Diag_04205, RS_Diag_04189

⌈In case DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to calculated, the DM shall store freeze frames numbered consecutively starting with 1 in their chronological order. If the parameter is set to configured, the DM shall store the records based on the DiagnosticFreezeFrame.recordNumber configuration parameters of the respective freeze frames.⌋

[SWS_DM_00152] Number of snapshot records for a DTC

Upstream requirements: RS_Diag_04205, RS_Diag_04190

⌈In case DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to calculated, the number of snapshot record the DM is able to store for a DTC shall be determined by the DiagnosticTroubleCodeProps.maxNumberFreezeFrameRecords configuration parameter. In case DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to configured, the number of snapshot recordss is determined by the number of DiagnosticFreezeFrames configured for a DTC.⌋

Note that different snapshot records represent different snapshots collected in different points in time.

[SWS_DM_01276] Triggering for snapshot record storage (calculated, maxNumberFreezeFrameRecords = 1)

Upstream requirements: RS_Diag_04205, RS_Diag_04127

⌈If DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to calculated and DiagnosticTroubleCodeProps.maxNumberFreezeFrameRecords is configured to 1, the DM shall collect and store the snapshot record only for the first transition of UDS DTC status bit 'testFailed' from '0' to '1'.⌋

[SWS_DM_01277] Triggering for snapshot record storage (calculated, maxNumberFreezeFrameRecords > 1)

Upstream requirements: RS_Diag_04205, RS_Diag_04127

⌈If DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration is set to calculated and DiagnosticTroubleCodeProps.maxNumberFreezeFrameRecords is configured to a value greater than 1, the DM shall collect and store the snapshot record on transition of UDS DTC status bit 'testFailed' from '0' to '1' consecutively in the snapshot records and in case all snapshot records are occupied only update the most recent snapshot record.⌋

[SWS_DM_01085] Triggering for snapshot record storage (configured, without update)

Upstream requirements: RS_Diag_04205, RS_Diag_04127

⌈If `DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration` is set to `configured`, `DiagnosticFreezeFrame.update` is set to 'False', the configured `DiagnosticFreezeFrame.trigger` is fulfilled and the snapshot is not yet stored, the `DM` shall collect and store the `snapshot record`.⌋

[SWS_DM_01086] Triggering for snapshot record storage (configured, with update)

Upstream requirements: RS_Diag_04205, RS_Diag_04127

⌈If `DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration` is set to `configured`, `DiagnosticFreezeFrame.update` is set to 'True' and the configured `DiagnosticFreezeFrame.trigger` is fulfilled, the `DM` shall collect and store, respectively update the `snapshot record`.⌋

[SWS_DM_01087] Snapshot record layout

Upstream requirements: RS_Diag_04205

⌈If any `snapshot record` storage trigger occures, the `DM` shall capture the data defined by `DiagnosticTroubleCodeProps.snapshotRecordContent`. Each referenced `DiagnosticDataIdentifier` shall be captured via the `PortPrototype` configured for these `DIDs`.⌋

[SWS_DM_00894] Notification event upon `snapshot record` updates

Upstream requirements: RS_Diag_04148, RS_Diag_04091

⌈After the `DM` has captured and stored a new `snapshot record` or overwritten an existing `snapshot record` with new data and there is a registered update notification via the function `ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier`, the `DM` shall call this notifier for each `snapshot record` update.⌋

In case of

- deletion (7.3.4.4.5)
- aging (7.3.4.4.6)
- displacement (7.3.4.4.10)

the `DM` doesn't trigger the notification calls for updates.

[SWS_DM_01351] Consecutive registration of notifier with SetSnapshotRecordUpdatedNotifier()

Upstream requirements: RS_Diag_04183

⌈In case of a consecutive call of `ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier` of the corresponding `ara::diag::DTCInformation` instance, `DM` module shall overwrite the previous registered notifier.⌋

7.3.4.4.4.3 Storage of extended data

This section describes the configuration of and the access to extended data for a `DTC`.

[SWS_DM_00154] Number of extended data for a `DTC`

Upstream requirements: RS_Diag_04206, RS_Diag_04190

⌈The `DM` shall store zero or one extended data for a `DTC`. Extended data consists of `extended data records`. If at least one `DiagnosticTroubleCodeProps.extendedDataRecord` is configured for the corresponding `DTC`, the extended data shall be present in the event memory entry.⌋

Note that contrary to `snapshot records`, `extended data records` do not necessarily represent data collected in different points in time. Extended data consists of a configurable number of `extended data records`, which are all collected when the respective memory entry is created in the event memory. The update mechanism of `extended data records` is configurable.

An `extended data record` typically contains `DM internal` data information. This is represented as `DiagnosticDataElement` referenced from `DiagnosticProvidedDataMapping`, where the `dataProvider` defines the content of the `internal` data. Such data elements can only be used within the scope of an `extended data records`. The DEXT limits a use in other `DiagnosticDataElement` such as from `snapshot records` or `DiagnosticDataElement` from `DIDs`.

[SWS_DM_00155] Extended data record numeration

Upstream requirements: RS_Diag_04206, RS_Diag_04189

⌈Extended data record numbers shall always be determined by the configuration. The `DiagnosticExtendedDataRecord.recordNumber` configuration parameter defines the record number for each `extended data record`.⌋

[SWS_DM_00895] Triggering for extended data record storage and updates

Upstream requirements: RS_Diag_04206, RS_Diag_04127

⌈The data collection and storage of the `extended data record` shall be triggered by the `DiagnosticExtendedDataRecord.trigger`. Updating extended data

records after being first stored, shall be configurable with the DiagnosticExtendedDataRecord.update configuration parameter. The data layout of extended data record is defined by the order of DiagnosticExtendedDataRecord.recordElement. Each DiagnosticDataElement shall be captured in its order via the corresponding read function instance for Typed DataElement: namespacelistdataelement::dataelementinterfacename::Read.⌋

7.3.4.4.4.4 Internal statistical data elements in EDRs

The DM module provides the ability to map internal data elements, like e.g. aging counter, occurrence counter, ... (see [SWS_DM_01565] for the full list) to a specific dataElement contained in a DiagnosticExtendedDataRecord.

If a DM - internal DiagnosticDataElement is mapped to an extended data record by configuration, this information can be requested by the UDS service 0x19 ReadDTCInformation - SubFunction 0x06 reportDTCExtendedDataRecordByDTCNumber (7.3.2.8.9.5).

The internal data elements with context "DEM" in [SWS_DM_01565] are not additionally stored or frozen in the extended data record when the event memory storage is triggered.

[SWS_DM_00949] Generation and usage of internal DiagnosticDataElements

Upstream requirements: RS_Diag_04127, RS_Diag_04190

⌈If an internal DiagnosticDataElement with context "DEM" gets used when the error memory is read out via diagnostic communication, the DM shall use the current value of that internal data element at the time when the error memory is read out.⌋

[SWS_DM_00950] Configuration of DTC priority as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_PRIORITY, the DM shall set the value of this internal data element to the DTC priority assigned by DiagnosticTroubleCodeProps.priority for this DTC.
The length of this internal data element is one byte.⌋

[SWS_DM_00921] Configuration of Error Memory Overflow Indication as extended data record

Upstream requirements: RS_Diag_04093, RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_OVFLIND, the DM shall set the value of this internal data element to the DM-internal value for event memory overflow to which the related DTC belongs to and map it:

- "0" = False, in case no event memory overflow was detected.
- "1" = True, in case an event memory overflow was detected.

The length of this internal DiagnosticDataElement is one byte.⌋

For more details, see also 7.3.4.4.9 “Event memory overflow”.

[SWS_DM_00951] Configuration of DTC "current FDC" as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_CURRENT_FDC, the DM shall report the internal value of the current Fault Detection Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord’s DiagnosticDataElement.
The length of this internal data element is one byte.⌋

The value translation from the internal debouncing mechanisms to the FDC is defined in [SWS_DM_00017] and [SWS_DM_00030].

[SWS_DM_00952] Configuration of DTC "max. FDC since clear" as extended data record

Upstream requirements: RS_Diag_04068, RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_MAX_FDC_SINCE_LAST_CLEAR, the DM shall report the internal value of the current maximum Fault Detection Counter since last clear of the contextual DTC in the respective DiagnosticExtendedDataRecord’s DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_00953] Configuration of DTC "max. FDC current cycle" as extended data record

Upstream requirements: RS_Diag_04127, RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_MAX_FDC_DURING_CURRENT_CYCLE, the DM shall report the internal value of the current maximum Fault Detection Counter during the current operation cycle of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_00954] Configuration of DTC "occurrence counter" as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_OCCCTR, the DM shall report the internal value of the current occurrence counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

For Event occurrence see 7.3.4.1.8.

[SWS_DM_00955] Configuration of DTC "aging counter up/down" as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_AGINGCTR_UPCNT or to DEM_AGINGCTR_DOWNCNT, the DM shall report the internal value of the current aging counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement based on [SWS_DM_00956] or [SWS_DM_00957].⌋

For Aging see 7.3.4.4.6.

[SWS_DM_00956] Configuration of DTC "aging counter up" as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider is set to DEM_AGINGCTR_UPCNT, the DM shall map the internal aging counter in such a way that a counting-up mode from '0' to the DiagnosticAging.threshold value is created according to ISO 14229-1[1], Annex D.⌋

[SWS_DM_00957] Configuration of DTC "aging counter down" as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider is set to DEM_AGINGCTR_DOWNCNT, the DM shall map the internal aging counter in such a way that a counting-down mode from the DiagnosticAging.threshold value to '0' is created.⌋

[SWS_DM_00958] Default value for DTC "aging counter up" if aging is not allowed

Upstream requirements: RS_Diag_04190

⌈If the element DiagnosticTroubleCodeProps.aging does not exist and the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_AGINGCTR_UPCNT, the DM shall set the respective DiagnosticExtendedDataRecord's DiagnosticDataElement value to '0'.⌋

[SWS_DM_00959] Default value for DTC "aging counter down" if aging is not allowed

Upstream requirements: RS_Diag_04190

⌈If the element DiagnosticTroubleCodeProps.aging does not exist and the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_AGINGCTR_DOWNCNT, the DM shall set the respective DiagnosticExtendedDataRecord's DiagnosticDataElement value to DiagnosticAging.threshold if configured or '255' otherwise.⌋

[SWS_DM_CONSTR_00960] No support for DEM_AGINGCTR_UPCNT_FIRST_ACTIVE

Upstream requirements: RS_Diag_04133

⌈"DEM_AGINGCTR_UPCNT_FIRST_ACTIVE" for the configuration parameter DiagnosticProvidedDataMapping.dataProvider of a DiagnosticParameter.dataElement shall not be supported by the DM.⌋

[SWS_DM_00961] Configuration of a DTCs significance as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_SIGNIFICANCE, the DM shall set the value of this internal data element to the DTCs significance assigned by DiagnosticTroubleCodeProps.significance for this DTC and map it:

- "0" = occurence.

- "1" = fault.

The length of this internal DiagnosticDataElement is one byte.⌋

[SWS_DM_00962] Configuration of a DTCs Failed Operation Cycles as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_FAILED_CYCLES, the DM shall report the internal value of the current Failed Operation Cycles Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_00963] Configuration of a DTCs failed operation Cycles Since First Failed as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_CYCLES_SINCE_FIRST_FAILED, the DM shall report the internal value of the current Operation Cycles Since First Failed Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_00964] Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_CYCLES_SINCE_LAST_FAILED, the DM shall report the internal value of the current Operation Cycles Since Last Failed Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_01954] Configuration of a DTCs failed operation Cycles Tested Since Last Failed as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_CYCLES_TESTED_SINCE_LAST_FAILED, the DM shall report the internal value of the current Operation Cycles Tested Since Last Failed Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement.
The length of this internal data element is one byte.⌋

[SWS_DM_01955] Configuration of a DTCs failed operation Cycles Tested Since First Failed as extended data record

Upstream requirements: RS_Diag_04190

⌈If the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement is set to DEM_CYCLES_TESTED_SINCE_FIRST_FAILED, the DM shall report the internal value of the current Operation Cycles Tested Since First Failed Counter of the contextual DTC in the respective DiagnosticExtendedDataRecord's DiagnosticDataElement. The length of this internal data element is one byte.⌋

[SWS_DM_01956] Latching of Internal Data Element DEM_CYCLES_TESTED_SINCE_LAST_FAILED

Upstream requirements: RS_Diag_04190

⌈The DM shall latch internal data element DEM_CYCLES_TESTED_SINCE_LAST_FAILED if it has reached 255 and no longer increment it⌋

[SWS_DM_01957] Latching of Internal Data Element DEM_CYCLES_TESTED_SINCE_FIRST_FAILED

Upstream requirements: RS_Diag_04190

⌈The DM shall latch internal data element DEM_CYCLES_TESTED_SINCE_FIRST_FAILED if it has reached 255 and no longer increment it.⌋

[SWS_DM_01958] Availability of internal data element Internal Data Element DEM_CYCLES_TESTED_SINCE_LAST_FAILED

Upstream requirements: RS_Diag_04190

⌈The DM shall support the internal data element DEM_CYCLES_TESTED_SINCE_LAST_FAILED only for stored events.⌋

[SWS_DM_01959] AAvailability of internal data element Internal Data Element DEM_CYCLES_TESTED_SINCE_FIRST_FAILED

Upstream requirements: RS_Diag_04190

⌈The DM shall support the internal data element DEM_CYCLES_TESTED_SINCE_FIRST_FAILED only for stored events.⌋

7.3.4.4.5 Clearing DTCs

Clearing a DTC or a DTC group is the ability of the DM to reset the UDS DTC status byte of each DTC and deleting DTC assigned snapshot records, extended data records and further DTC-related data.

[SWS_DM_00116] Clearing a DTC group

Upstream requirements: RS_Diag_04117

⌈When the DM is about to clear a DTC group it shall apply the same clear operation process as for a single DTC on all the DTCs of the DTC group which is cleared.⌋

[SWS_DM_00117] Clearing a DTC

Upstream requirements: RS_Diag_04117

⌈When the DM is about to clear a DTC it shall reset the event and UDS DTC status byte and clear the snapshot records and extended data records stored for this DTC and its DTC-related data.⌋

7.3.4.4.5.1 Locking of the DTC clearing process by a client

The DM supports more than one Diagnostic Clients as described in section 7.3.2.1.1. All configured clients can simultaneously send a ClearDTC diagnostic request. This chapter describes the DM behavior in this situations.

[SWS_DM_00144] Parallel clearing DTCs in different DiagnosticMemoryDestination

Upstream requirements: RS_Diag_04117

⌈The DM shall support parallel clearing of DTCs if the target of the clear DTC operation is a different DiagnosticMemoryDestination.⌋

[SWS_DM_00145] Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination

Upstream requirements: RS_Diag_04117

⌈If a Diagnostic Client is clearing the DTCs of a DiagnosticMemoryDestination the DM shall lock the clear DTC operation for all other clients requesting to clear the DTCs of the same DiagnosticMemoryDestination.⌋

[SWS_DM_00146] Unlock clear DTC operation for one DiagnosticMemoryDestination

Upstream requirements: RS_Diag_04117

⌈After the DM has finished the clear DTC operation, it shall unlock the clear DTC operation for this DiagnosticMemoryDestination.⌋

[SWS_DM_00147] Behavior while trying to clear DTCs on a locked DiagnosticMemoryDestination

Upstream requirements: RS_Diag_04117

⌈If the DM is requested to clear DTCs of a DiagnosticMemoryDestination and the DM has locked this DiagnosticMemoryDestination for clearing DTCs according to [SWS_DM_00144], the DM shall refuse the second clear DTC operation and shall return a NRC 0x22 (ConditionsNotCorrect).⌋

7.3.4.4.5.2 ClearConditions

In certain situations it is desirable to avoid that a DTC is cleared from the event memory. DiagnosticClearConditions are mapped to DTCs by DiagnosticTroubleCodeUdsToClearConditionGroupMappings.

[SWS_DM_00896] Handling of DiagnosticClearConditions

Upstream requirements: RS_Diag_04117

⌈If any of the clear conditions mapped to the DTC to be cleared are not fulfilled by a call of the function ara::diag::Condition::SetCondition with the value kConditionFalse, the clear is forbidden. Otherwise (all of the clear conditions mapped to the DTC are fulfilled) the clear is allowed.⌋

The effect of a forbidden clear DTC operation is described in the requirements below:

[SWS_DM_00123] Block clearing of UDS DTC status byte during a clear DTC operation

Upstream requirements: RS_Diag_04117

⌈If the DM is requested to clear a DTC with a forbidden clear according to [SWS_DM_00896] and a DiagnosticEventToTroubleCodeUdsMapping exists with a mapping from this DTC to an event and the event has DiagnosticEvent.clearEventAllowedBehavior set to noStatusByteChange, the DM shall not change the UDS DTC status byte.⌋

[SWS_DM_00124] Limited clearing of UDS DTC status byte during a clear DTC operation

Upstream requirements: RS_Diag_04117

⌈If the DM is requested to clear a DTC with a forbidden clear according to [SWS_DM_00896] and a DiagnosticEventToTroubleCodeUdsMapping exists with a mapping from this DTC to an event and the event has DiagnosticEvent.clearEventAllowedBehavior set to onlyThisCycleAndReadiness, the DM shall set the following UDS DTC status bits:

- Bit 1 TestFailedThisOperationCycle to '0'
- Bit 4 TestNotCompletedSinceLastClear to '1'
- Bit 5 TestFailedSinceLastClear to '0'
- Bit 6 TestNotCompletedThisOperationCycle to '1'

and leave all other bits unchanged.⌋

[SWS_DM_00121] Forbidden clearing of `snapshot records` and `extended data records`

Upstream requirements: RS_Diag_04117

⌈If the `DM` is requested to clear a `DTC` with a forbidden clear according to [SWS_DM_00896] the `DM` shall leave all `snapshot records` and `extended data records` for this `DTC` unchanged.⌋

7.3.4.4.5.3 `DTC` clearing triggered by application

Besides the UDS request ClearDiagnosticInformation according to section 7.3.2.8.8.1 the `DM` supports the use case that the fault memory is cleared by an application call. One of the use cases is clearing of `user-defined event memory`. This could be realized using a dedicated diagnostic routine service, whose application is in charge of the clearing process.

The clear `DTC` operation itself is semantically identical, independent if triggered via diagnostic service or application method call. All requirements for clear `DTC` apply in either case.

[SWS_DM_00897] Usage of ClearDTC Interface

Upstream requirements: RS_Diag_04194

⌈If the function `ara::diag::DTCInformation::Clear` is called, the `DM` shall clear the `DTC` or `DTC group` provided in the functions parameter `dtcGroup`. The clear `DTC` shall clear the fault memory associated to the instance of the `ara::diag::DTCInformation` class only.⌋

[SWS_DM_00898] ClearDTC call on invalid `DTC` or `DTC group`

Upstream requirements: RS_Diag_04194

⌈If the function `ara::diag::DTCInformation::Clear` is called and the functions parameter `dtcGroup` has no matching configured `DTC group` according to [SWS_DM_00064] or has no matching configured `DTC` by `DiagnosticTroubleCodeUds.udsDtcValue`, the `DM` shall trigger the error `kWrongDtc` for that function call and the `DM` shall return without any further action.⌋

[SWS_DM_00899] ClearDTC called while another clear operation is in progress

Upstream requirements: RS_Diag_04194

⌈If the function ara::diag::DTCInformation::Clear is called and another clear DTC operation is currently in progress, the DM shall trigger the error kBusy.⌋

[SWS_DM_00900] ClearDTC processing in case of memory errors

Upstream requirements: RS_Diag_04194

⌈If the function ara::diag::DTCInformation::Clear is called and the DM receives physical memory errors upon its access to the Non-volatile Memory and thus cannot guarantee that the clear operation was done successfully, the DM shall trigger the error kMemoryError.⌋

[SWS_DM_00901] Possible failure of ClearDTC

Upstream requirements: RS_Diag_04194

⌈If the function ara::diag::DTCInformation::Clear is called and the clear operation fails due to the reasons according to [SWS_DM_00122], the DM shall trigger the error kFailed.⌋

7.3.4.4.6 Aging

A stored DTC can age in terms of reaching a DiagnosticAging.threshold value of passed operation cycles, specified by the vendor, where no failed tests have been reported by a monitoring application. The amount of operation cycles, where these non-failed reports occur is called the Aging counter. After the threshold is reached, the DTC is cleared from the event memory.

[SWS_DM_00237] Aging

Upstream requirements: RS_Diag_04133

⌈If the DiagnosticTroubleCodeProps.aging exists, The DM shall support Aging for all DTCs referring to this DiagnosticTroubleCodeProps.aging.⌋

[SWS_DM_00238] Aging and healing

Upstream requirements: RS_Diag_04133

⌈If an indicator is configured for the corresponding event, the process of Aging (counting of Aging counter) shall be started only after the healing (according to [SWS_DM_00224]) is completed ('warningIndicatorRequested' bit is set to 0).⌋

[SWS_DM_00239] Aging counter

Upstream requirements: RS_Diag_04133

⌈The DM shall support an Aging counter for each event memory entry.⌋

Note that this counter shall be available as internal data element of extended data records.

The implementation of the internal aging counter mechanism is independent from the configuration parameter DiagnosticProvidedDataMapping.dataProvider of the corresponding DiagnosticParameter.dataElement set to DEM_AGINGCTR_UPCNT or to DEM_AGINGCTR_DOWNCNT. Only for reading the internal data element, a mapping as defined in [SWS_DM_00956] and [SWS_DM_00957] is applied.

[SWS_DM_00240] Processing the Aging counter

Upstream requirements: RS_Diag_04133

⌈The DM shall only allow processing the Aging counter if the related DTC is stored in the event memory, testFailedThisOperationCycle bit is set to '0' and healing, according to [SWS_DM_00238], is fulfilled.⌋

[SWS_DM_00241] Aging cycle and threshold

Upstream requirements: RS_Diag_04133

⌈The Aging shall be calculated based on the referred DiagnosticOperationCycle via the reference DiagnosticAging.agingCycle.⌋

[SWS_DM_01264] DiagnosticAging.threshold reached

Upstream requirements: RS_Diag_04133

⌈The DiagnosticAging.threshold defines the number of Aging cycles until Aging. If the threshold is reached, the event memory entry shall be deleted (aged) from the event memory including the snapshot records and extended data records belonging to that aged DTC.⌋

[SWS_DM_01265] Aging requires tested cycles only

Upstream requirements: RS_Diag_04133

⌈If an operation cycle is restarted and DiagnosticMemoryDestination.agingRequiresTestedCycle is set to True, the DM shall increment/decrement the aging counter, if the events status has 'testNotCompletedThisOperationCycle' bit and 'testFailedThisOperationCycle' set to 0.⌋

[SWS_DM_01953] Aging for untested cycles

Upstream requirements: RS_Diag_04133

⌈If an operation cycle is restarted and DiagnosticMemoryDestination.agingRequiresTestedCycle is set to False, the DM shall increment/decrement the aging counter, if the events status has 'testFailedThisOperationCycle' set to 0.⌋

[SWS_DM_00243] Aging-related UDS DTC status byte processing

Upstream requirements: RS_Diag_04140

⌈If a DTC is aged, the DM shall set the following UDS DTC status bits to 0:

- 'confirmedDTC' unconditionally
- 'testFailedSinceLastClear' conditionally, if statusBitHandlingTestFailedSinceLastClear is set to statusBitAgingAndDisplacement

⌋

After DTC - Aging all event related data for this DTC is cleared. If an event mapped to this DTC fails again a new event memory entry with initial values is created.

[SWS_DM_01576] Behavior of not configured DiagnosticMemoryDestination.agingRequiresTestedCycle

Upstream requirements: RS_Diag_04133

⌈If the optional parameter DiagnosticMemoryDestination.agingRequiresTestedCycle is not configured, the DM shall use a default value of 'FALSE' for that parameter.⌋

[SWS_DM_01577] Behavior of not configured DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear

Upstream requirements: RS_Diag_04140

⌈If the optional parameter DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear is not configured, the DM shall use a default value of 'statusBitNormal' for that parameter.⌋

7.3.4.4.7 NumberOfStoredEntries

[SWS_DM_00902] NumberOfStoredEntries

Upstream requirements: RS_Diag_04109

⌈If the function ara::diag::DTCInformation::GetNumberOfStoredEntries is called, the DM shall return the number of event memory entries (DTCs) currently

stored in this event memory. An update notification shall be sent to the function registered via ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier whenever the value of NumberOfStoredEntries has changed.⌋

[SWS_DM_01352] Consecutive registration of notifier with SetNumberOfStoredEntriesNotifier()

Upstream requirements: RS_Diag_04109

⌈In case of a consecutive call of ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier of the corresponding ara::diag::DTCInformation instance, DM module shall overwrite the previous registered notifier.⌋

7.3.4.4.8 Active / Passive Status of Events

If an event gets qualified as failed, it becomes "active". If the event gets qualified as passed, it becomes "passive". This status can be derived from the UDS DTC status byte. The UDS DTC status byte is persistently stored over power cycles. "**Event active**" equals to 'TestFailed = 1' and "**event passive**" equals to 'TestFailed = 0'.

7.3.4.4.9 Event memory overflow

An event memory is considered to be full in case that already <DiagnosticMemoryDestination.maxNumberOfEventEntries> are stored in this event memory. If in this situation a new event needs to be stored in this event memory, an event memory overflow occurs and error information got lost.

An event memory overflow can happen to primary and user-defined event memories.

[SWS_DM_00922] Persistent storage for event memory overflow information

Upstream requirements: RS_Diag_04093

⌈The DM module shall store and provide the event memory overflow information persistently for each of the configured event memories separately.⌋

[SWS_DM_00923] Event memory overflow set condition

Upstream requirements: RS_Diag_04093

⌈If there exists already <maxNumberOfEventEntries> in one of the configured DM event memories and there is an attempt to store an additional entry to this event memory, the DM module shall from then on return true on calling the function ara::diag::DTCInformation::GetEventMemoryOverflow for this event memory instance of ara::diag::DTCInformation.⌋

This overflow indication can be used to trigger further internal behavior of the DM module (e.g. displacement strategy). Furthermore, it can be used for additional fault analysis in workshops in case this overflow information is used in a DTC extended data record.

[SWS_DM_00924] Event memory overflow reset condition

Upstream requirements: RS_Diag_04093

⌈If there never happened an overflow before (compare to [SWS_DM_00923]) or if the clear of all DTCs was executed for a specific event memory, the DM shall from then on return false on calling the function ara::diag::DTCInformation::GetEventMemoryOverflow for this event memory instance.⌋

In case of aging and deleting single DTCs, the overflow indication of the event memory is not reset.

[SWS_DM_00925] Event memory overflow notifier on occurence

Upstream requirements: RS_Diag_04093

⌈If there exists already <maxNumberOfEventEntries> in one of the configured DM event memories and there is an attempt to store an additional entry to this event memory, the DM shall each time call the corresponding overflow notification function for that event memory, which was registered via the function ara::diag::DTCInformation::SetEventMemoryOverflowNotifier for this event memory instance of ara::diag::DTCInformation, with the parameter value set to true.⌋

[SWS_DM_00926] Event memory overflow notifier on clear

Upstream requirements: RS_Diag_04093

⌈If an overflow has occurred, as specified in [SWS_DM_00923]), for a particular event memory, the DM shall, after the next execution of clear all DTCs for that particular event memory, call the corresponding overflow notification function for that event memory, which was registered via the function ara::diag::DTCInformation::SetEventMemoryOverflowNotifier for this event memory instance, with the parameter value set to false.⌋

[SWS_DM_01354] Consecutive registration of notifier with SetEventMemoryOverflowNotifier()

Upstream requirements: RS_Diag_04093

⌈In case of a consecutive call of ara::diag::DTCInformation::SetEventMemoryOverflowNotifier of the corresponding ara::diag::DTCInformation instance, DM module shall overwrite the previous registered notifier.⌋

7.3.4.4.10 Event memory entry displacement

Displacement is applied in case the event memory has already reached the maximum allowed number of stored entries and a further new event memory entry shall be stored.

In this case the decision between

- displacing an already earlier stored event memory entry

or

- discarding the new reported event

needs to be taken.

Displacement means, that the least significant, already existing event memory entry is replaced by a new reported and more significant event, which needs to be stored. During displacement, the least significant entry gets lost.

If there is no maximum allowed number of entries for a specific event memory or if the maximum allowed number is configured to cover all possible events, no displacement will occur.

In the following, the expression "overflow situation" is used for the condition that the event memory was already full, i.e. <maxNumberOfEventEntries> were already stored in that event memory and now a new entry needs to be added to that event memory.

[SWS_DM_00927] Disabled displacement

Upstream requirements: RS_Diag_04118

⌈If DiagnosticMemoryDestination.eventDisplacementStrategy selects none and an overflow situation occurred in that particular event memory, the DM shall discard the new reported event.⌋

[SWS_DM_00928] Priority and occurrence based displacement

Upstream requirements: RS_Diag_04118, RS_Diag_04105

⌈If DiagnosticMemoryDestination.eventDisplacementStrategy selects prioOcc and an overflow situation occurred in that particular event memory, the DM shall

- **step 1**: search through that event memory for entries that
  - have the lowest priority value in that event memory

    AND
  - have a lower priority than the new entry.

- **step 2**: Out of that list the DM shall select the chronologically oldest occurred memory entry for the displacement operation.

⌋

For strategy prioOcc there is no displacement for equal or higher priority event memory entries. The UDS DTC status bits are also not considered.

[SWS_DM_00929] Displacement strategy "full"

Upstream requirements: RS_Diag_04118, RS_Diag_04105

⌈If DiagnosticMemoryDestination.eventDisplacementStrategy selects full and an overflow situation occurred in that particular event memory, the DM module shall perform the following selection sequence by combination of the different displacement criteria, listed by their descending importance:

- **1) Priority** (compare [SWS_DM_00916]): search through that event memory for entries that have the lowest priority value in that event memory.
- **2) Active** / **Passive Status**: out of the above filtered selection from 1): search for events in the following order:
  - **a.)**: If the lowest priority in the event memory is less than the priority of the new event:
    - **i.)**: In the first place, the DM shall select Passive events .
    - **ii.)**: In case no Passive events are available, the DM shall select all events from the above filtered criteria (independent from the UDS DTC status bits).
  - **b.)**: If the lowest priority in the event memory is equal to the priority of the new event:
    - **i.)**: In the first place, the DM shall select Passive events.
    - **ii.)**: In case no Passive events are available, the DM shall select events with UDS DTC status bit "TestNotCompletedThisOperationCycle" is set.
- **3) Oldest entry**: If the selection from the above criteria results in one or more event entries, the DM shall select the chronologically oldest occurred event memory entry for the displacement operation.

⌋

Details about Active / Passive Status are specified in 7.3.4.4.8 “Active / Passive Status of Events”

For strategy `full` there is no `displacement` for active (testCompletedThisOperationCycle) event memory entries with equal priority or for higher priority event memory entries.

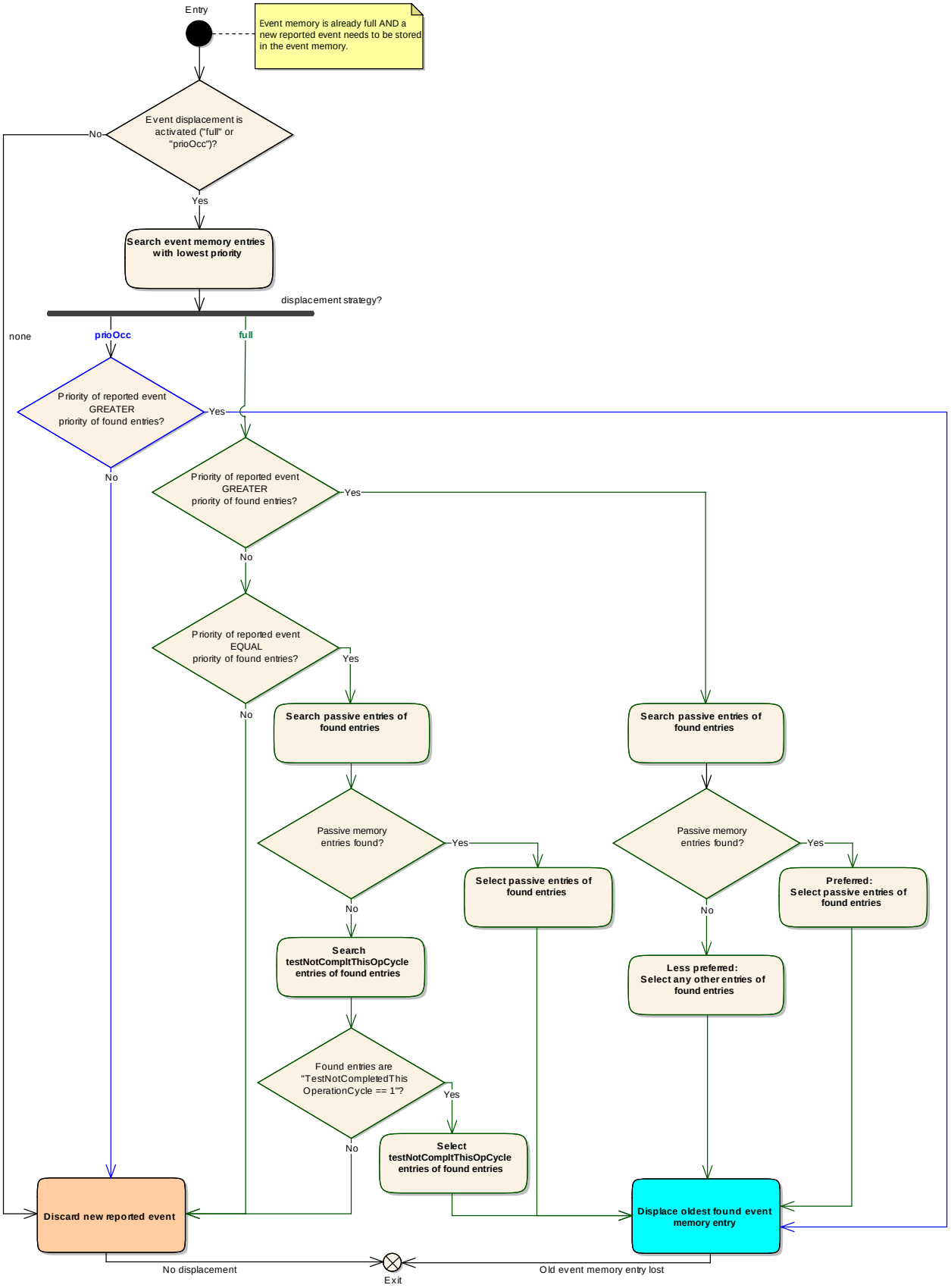

Figure 7.8: Combined displacement criteria processing

[SWS_DM_00930] Displacement operation

Upstream requirements: RS_Diag_04118

⌈If an event memory entry for displacement is identified as specified in [SWS_DM_00928] or [SWS_DM_00929], the DM module shall remove this old event memory entry from the event memory and add the new reported event to the memory.⌋

[SWS_DM_01569] Configurable reset of the pendingDTC bit in case of displacement

Upstream requirements: RS_Diag_04067, RS_Diag_04118

⌈If an event memory entry was removed during displacement and the configuration parameter DiagnosticCommonProps.resetPendingBitOnOverflow is set to "true", the DM module shall reset the UDS DTC status bit 2 / 'pendingDTC' to 0.⌋

[SWS_DM_00932] UDS DTC status bit 3 / 'ConfirmedDTC' after displacement

Upstream requirements: RS_Diag_04067, RS_Diag_04118

⌈If an event memory entry was removed during displacement AND the configuration parameter DiagnosticCommonProps.resetConfirmedBitOnOverflow is set to "true", the DM module shall reset the UDS DTC status bit 3 / 'ConfirmedDTC' to '0'.⌋

[SWS_DM_00933] UDS DTC status bit 5 / 'testFailedSinceLastClear' after displacement

Upstream requirements: RS_Diag_04067, RS_Diag_04118

⌈If an event memory entry was removed during displacement AND the configuration parameter DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear is set to statusBitAgingAndDisplacement AND DiagnosticCommonProps.resetConfirmedBitOnOverflow is set to "true", the DM shall reset the UDS DTC status bit 5 / 'testFailedSinceLastClear' to '0'.⌋

[SWS_DM_00934] Condition for discarding the new event

Upstream requirements: RS_Diag_04118

⌈If an overflow situation occurred and no event memory entry for displacement was identified as specified in [SWS_DM_00928] and [SWS_DM_00929], the DM module shall discard the storage request for the new reported event.⌋

7.3.4.4.11 Reporting order of event memory entries

[SWS_DM_00981] Conditions of status based reporting order

Upstream requirements: RS_Diag_04195

⌈Upon requests of the following sub-functions from UDS service ID 0x19 as shown in [SWS_DM_01566], the DM module shall report DTCs in the chronological order of the event storage (compare memoryEntryStorageTrigger), if:

- the DTCStatusMask parameter in the UDS request message has the UDS DTC status bit 'pending DTC' or 'confirmed DTC' bit or both bits set and
- all other UDS DTC status bits of the DTCStatusMask parameter in the UDS request message are set to false and
- resetConfirmedBitOnOverflow is set to true.

⌋

[SWS_DM_01566] Subfunctions of 0x19 / ReadDTCInformation with chronolicical reporting order

Upstream requirements: RS_Diag_04195

⌈

SSID	Name
0x02	reportDTCByStatusMask(DTCStatusMask)
0x17	reportUserDefMemoryDTCByStatusMask(DTCStatusMask, MemorySelection)

Table 7.3

⌋

[SWS_DM_00982] Reporting order direction

Upstream requirements: RS_Diag_04195

⌈If the DM module is requested to report in chronological order as specified in [SWS_DM_00981], the most recent event memory entry shall be reported at first.⌋

7.3.5 Required Configuration

The Autosar Diagnostic Extract Template (DEXT) [3] is used for the DM configuration. By design this format is made as exchange format between the tools in the diagnostic workflow, in different steps data is added. To accommodate the fact that data is incomplete and refined in a later step, the DEXT [3] allows most of the elements to be optional and added at a later point in time. However at the point in time, when the DEXT [3] is used to configure the DM, a certain minimum content is required. In this

chapter a loose list of DEXT [3] constraints is given. The mentioned elements need to be present so that the DM can be configured. Also the reaction on such missing elements is implementation specific, it is stated that the DM will not be able to behave as described in the document. A possible but not mandatory reaction is to refuse the DM generation at all and forcing the user to provide complete data.

[SWS_DM_CONSTR_00168] Required operation cycles for diagnostic events

Upstream requirements: RS_Diag_04178

⌈Each DiagnosticEvent requires exactly one DiagnosticEventToOperationCycleMapping referencing the diagnosticEvent and one DiagnosticOperationCycle.⌋

[SWS_DM_CONSTR_00206] Supported format for data identifier for VINDataIdentifier

Upstream requirements: RS_Diag_00026

⌈A DiagnosticDataIdentifier with representsVin set to true, requires that it aggregates only one DiagnosticParameter which itself aggregates a DiagnosticDataElement having a 17 byte uint8 array as baseType.⌋

7.3.6 Diagnostic Data Management

In various situations, the Diagnostic Server instance facilitates reading or writing of particular diagnostic data. One needs to distinguish between internal and external diagnostic data. By definition, internal data is managed by the Diagnostic Server instance itself, and external data is managed by external applications. In the latter case, communication between Diagnostic Server instance and the external application takes place via Service Interfaces. There are several Service Interfaces defined concerning diagnostic data.

The purpose of this chapter is to describe the supported use-cases for handling diagnostic data and the way how to configure each use-case within the DEXT.

Recall that a DiagnosticDataIdentifier is composed of DiagnosticParameters each of which aggregates a single DiagnosticDataElement. In different use cases, it is required to manage diagnostic data either on the level of DiagnosticDataIdentifier or on the fine granular level of DiagnosticDataElements.

7.3.6.1 Internal and External Diagnostic Data Elements

A DiagnosticDataElement is called internal if there exists a DiagnosticProvidedDataMapping referencing this DiagnosticDataElement, otherwise it is called an external DiagnosticDataElement.

[SWS_DM_01565] gives a list of the supported internal DiagnosticDataElements, where

Data Provider refers to the NameToken defined by the attribute dataProvider of the associated DiagnosticProvidedDataMapping,

Content describes the actual content of the data,

Format describes the data format of the DiagnosticDataElement.

Context defines the exclusive context in which this DiagnosticDataElement is defined (if applicable). For "DEM" see Diagnostic Event Management. For "DCM" see Diagnostic Communication Management.

[SWS_DM_01565] Supported internal DiagnosticDataElements

Upstream requirements: RS_Diag_04097

⌈

Data Provider	Content	Format	Context
DEM_AGINGCTR_DOWNCNT	Down-counting aging counter of contextual DTC	1 byte	DEM
DEM_AGINGCTR_UPCNT	Up-counting aging counter of contextual DTC	1 byte	DEM
DEM_CURRENT_FDC	Fault Detection Counter of contextual DTC	1 byte	DEM
DEM_CYCLES_SINCE_FIRST_FAILED	Operation Cycle Counter of contextual DTC– Cycles since first failed	1 byte	DEM
DEM_CYCLES_SINCE_LAST_FAILED	Operation Cycle Counter of contextual DTC – Cycles since last failed	1 byte	DEM
DEM_FAILED_CYCLES	Operation Cycle Counter of contextual DTC – Failed cycles	1 byte	DEM
DEM_MAX_FDC_DURING_CURRENT_CYCLE	Fault Detection Counter maximum value during current operation cycle of contextual DTC	1 byte	DEM
DEM_MAX_FDC_SINCE_LAST_CLEAR	Fault Detection Counter maximum value since last clear of contextual DTC	1 byte	DEM
DEM_OCCCTR	Occurrence counter of contextual DTC	1 byte	DEM
DEM_OVFLIND	Overflow indication of contextual DTCs error memory(0 = False, 1 = True)	1 byte	DEM
DEM_SIGNIFICANCE	Event significance of contextual DTC (refer to DemDTCSignificance) (0 = OCCURRENCE, 1 = FAULT)	1 byte	DEM
DEM_PRIORITY	Priority of the contextual DTC	1 byte	DEM
DCM_SESSION	Current session of contextual Diagnostic Conversation	1 byte	DCM
DCM_SECURITY_LEVEL	Current security level of contextual Diagnostic Conversation	1 byte	DCM
DEM_EVENT_ASSOCIATED_IDENTIFICATION	Represents the static value associated to it by associatedEventIdentification	4 byte	DEM
DEM_CYCLES_TESTED_SINCE_LAST_FAILED	Tested cycles since last failed excluding test not complete. The counter latches at 255	1 byte	DEM

Data Provider	Content	Format	Context
DEM_CYCLES_TESTED_SINCE_FIRST_FAILED	Tested cycles since first failed excluding test not complete. The counter latches at 255	1 byte	DEM

⌋

[SWS_DM_00393] Retrieving data for `internal DiagnosticDataElements`

Upstream requirements: RS_Diag_04097

⌈If `DM` requires to provide or store data configured as `internal DiagnosticDataElement` which is supported by the `Diagnostic Server instance` according to [SWS_DM_01565], then `DM` shall use the respective internally managed data value as defined in [SWS_DM_01565].⌋

[SWS_DM_CONSTR_00394] `Internal DiagnosticDataElements` are read-only

Upstream requirements: RS_Diag_04097

⌈A `DiagnosticDataIdentifier` referenced by a `DiagnosticWriteDataByIdentifier` service shall not contain any `internal DiagnosticDataElement`.⌋

An `internal DiagnosticDataElement` is called DCM-exclusive resp. DEM-exclusive if the context of the name token described in [SWS_DM_01565] is set accordingly. The implicit restriction of such `DiagnosticDataElements` to the context in which they are defined is made explicit in the following requirements. These requirements are formulated in a way that [SWS_DM_01565] might in future be extended by `internal DiagnosticDataElements` not restricted to exclusive use within a DCM resp. DEM context.

[SWS_DM_CONSTR_00395] Restriction on DEM-exclusive `DiagnosticDataElements`

Upstream requirements: RS_Diag_04097

⌈A `DiagnosticParameter` containing a DEM-exclusive `internal DiagnosticDataElement` shall not be contained in a `DiagnosticDataIdentifier` referenced by a `DiagnosticReadDataByIdentifier`, nor shall it be contained in a realization of `DiagnosticRoutineSubfunction`.⌋

[SWS_DM_CONSTR_00396] Restriction on DCM-exclusive `DiagnosticDataElements`

Upstream requirements: RS_Diag_04097

⌈A `DiagnosticParameter` containing a DCM-exclusive `internal DiagnosticDataElement` shall not be contained in a `DiagnosticDataIdentifier` referenced

by a DiagnosticDataIdentifierSet which is referenced by some DiagnosticTroubleCodeProps in the role of snapshotRecordContent, nor shall it be contained in a DiagnosticExtendedDataRecord.⌋

Note: The notion of internal and external is exclusively defined for DiagnosticDataElements and does not apply to DiagnosticDataIdentifier.

[SWS_DM_00905] Retrieving data for external DiagnosticDataElements

Upstream requirements: RS_Diag_04097

⌈If the Diagnostic Server instance is required to read data configured as external DiagnosticDataElement, then the Diagnostic Server instance shall utilize the associated RPortPrototype typed by the namespacelistdataelement::dataelementinterfacename class and call its namespacelistdataelement::dataelementinterfacename::Read function.⌋

Note: In general, there are multiple instances of namespacelistdataelement::dataelementinterfacename class available in the running system. Which instance to choose for the given request to read an external DiagnosticDataElement is part of system integration. Support for this integration is provided by DiagnosticMappings described in section 7.3.6.2.1.

7.3.6.2 Reading and Writing Diagnostic Data Identifier

The Diagnostic Server instance supports multiple ways to read or write diagnostic data defined as DiagnosticDataIdentifier:

- reading each DiagnosticDataElement contained in the DiagnosticDataIdentifier independently as described in section 7.3.6.1,
- reading or writing the DiagnosticDataIdentifier as a whole via the DataIdentifier diagnostic interface,
- reading or writing the DiagnosticDataIdentifier as a whole via the GenericService diagnostic interface.

The method to choose between these ways of data handling is by configuration of DiagnosticMappings referring to the DiagnosticDataIdentifier. This chapter describes the supported DiagnosticMappings and provides requirements on reading and writing DiagnosticDataIdentifier reflecting the short description above.

7.3.6.2.1 Supported Diagnostic Mappings

Details regarding the modeling of diagnostic mappings can be found in the TPS Manifest Specification [11].

7.3.6.2.2 Reading Diagnostic Data Identifier

[SWS_DM_00401] Reading Diagnostic Data Identifier on Data Element level

Upstream requirements: RS_Diag_04097

⌈If the `Diagnostic Server instance` is required to read data configured as `DiagnosticDataIdentifier` and all the `DiagnosticDataElements` aggregated in this `DiagnosticDataIdentifier` are referenced by `DiagnosticProvidedDataMapping` or `DiagnosticDataPortMapping`, then `Diagnostic Server instance` shall retrieve the data by reading data from each `DiagnosticDataElement` according to [SWS_DM_00393] and [SWS_DM_00905].⌋

[SWS_DM_00848] Reading Diagnostic Data Identifier by typed DataIdentifier interface

Upstream requirements: RS_Diag_04097

⌈If the `Diagnostic Server instance` is required to read data configured as `DiagnosticDataIdentifier` which is referenced by a `DiagnosticDataPortMapping`, then the `Diagnostic Server instance` shall use the `namespacelistdataidentifier::dataidentifierinterfacename` class and associated to the `DiagnosticDataIdentifier` for reading the data.⌋

[SWS_DM_01038] Reading Diagnostic Data Identifier by `ara::diag::GenericDataIdentifier` interface

Upstream requirements: RS_Diag_04097

⌈If the `Diagnostic Server instance` is required to read data configured as `DiagnosticDataIdentifier` which is referenced by a `DiagnosticDataPortMapping`, then the `Diagnostic Server instance` shall use the `ara::diag::GenericDataIdentifier` instance according to its `PortPrototype` mapping.⌋

[SWS_DM_00849] Reading Diagnostic Data Identifier by GenericUDSService interface

Upstream requirements: RS_Diag_04097

⌈If the Diagnostic Server instance is required to read data configured as DiagnosticDataIdentifier which is referenced by a DiagnosticServiceGenericMapping , then the Diagnostic Server instance shall use the instance of the ara::diag::GenericUDSService class referenced by the DiagnosticServiceGenericMapping and call its ara::diag::GenericUDSService::HandleMessage method with sid parameter set to 0x22 and requestData parameter set to the id of the DiagnosticDataIdentifier.⌋

The application realizing the ara::diag::GenericUDSService::HandleMessage is in the responsibility of a serialization/deserialization of UDS parameters. That means no data typed elements are provided as UDS input/output parameters.

7.3.6.2.3 Writing Diagnostic Data Identifier

[SWS_DM_00906] Writing Diagnostic Data Identifier by DataIdentifier interface

Upstream requirements: RS_Diag_04097

⌈If the Diagnostic Server instance is required to write data configured as DiagnosticDataIdentifier which is referenced by a DiagnosticDataPortMapping, then the Diagnostic Server instance shall use the ara::diag::GenericDataIdentifier instance and associated to the DiagnosticDataIdentifier for writing the data.⌋

[SWS_DM_01039] Writing Diagnostic Data Identifier by typed DataIdentifier interface

Upstream requirements: RS_Diag_04097

⌈If the Diagnostic Server instance is required to write data configured as DiagnosticDataIdentifier which is referenced by a DiagnosticDataPortMapping, then the Diagnostic Server instance shall use the namespacelist-dataidentifier::dataidentifierinterfacename according to its PortPrototype mapping.⌋

[SWS_DM_00908] Writing Diagnostic Data Identifier by GenericUDSService interface

Upstream requirements: RS_Diag_04097

⌈If the Diagnostic Server instance is required to write data configured as DiagnosticDataIdentifier which is referenced by a DiagnosticServiceGenericMapping, then the Diagnostic Server instance shall use the instance of the ara::diag::GenericUDSService class referenced by the DiagnosticServiceGenericMapping and call its ara::diag::GenericUDSService::HandleMessage with sid set to 0x2E and requestData set to the id of this DiagnosticDataIdentifier followed by the data to be written to this DiagnosticDataIdentifier.⌋

7.4 Functional cluster life-cycle

This chapter handles the diagnostics functional cluster lifecycle. It describes the behavior of the diagnostic daemon in several cases, like startup, shutdown or crash. In these cases the interaction with the application and diag API library is also described.

This chapter defines requirements, which are related to the lifecycle of the diagnostic functional cluster. It will describe the startup and shutdown procedure of the DM.

7.4.1 Startup

The startup procedure of the DM is treated as vendor-specific.

[SWS_DM_01571] Recovery of persisted data

Upstream requirements: RS_Main_00011

⌈The Diagnostic Server instance shall recover all relevant persisted data (e.g. data persisted during shutdown such as defined Dynamical Data Identifiers or Event and DTC status). The earliest point in time to recover this data is the startup of the Diagnostic Server instance. The latest point in time to recover this, is when the data is actually needed (lazy loading)⌋

7.4.2 Shutdown

In the following the required shutdown procedure of the DM daemon is defined.

It is expected, that during a synchronized/orchestrated shutdown applications, which are interacting with the DM, are already shutdown before the DM daemon is requested to be shutdown by the Execution Management.

[SWS_DM_01572] Graceful shutdown

Upstream requirements: RS_Main_00011

⌈Upon reception of SIGTERM the `Diagnostic Server instance` shall gracefully shutdown itself.⌋

[SWS_DM_01573] Stop all running `Transport Protocol Handlers`

Upstream requirements: RS_Main_00011

⌈During graceful shutdown, the `Diagnostic Server instance` shall call `apext::diag::uds_transport::UdsTransportProtocolHandler::Stop` on all started UdsTransportProtocolHandlers.⌋

[SWS_DM_01574] Write data to be persisted

Upstream requirements: RS_Main_00011

⌈During graceful shutdown, the `Diagnostic Server instance` shall persist all relevant data to be maintained over power down cycles.⌋

7.4.3 Daemon crash

In case the `DM` daemon crashes for some reason, it will as most as possible be restarted "silently", so that the adaptive applications can still run without a restart.

In chapter " 7.3.1.3" the caching for DEM related features is described to be able to re-sync after a daemon restart.

7.5 Reporting

7.5.1 Security Events

This section lists all security events defined by this functional cluster.

[SWS_DM_02014] Security events for Diagnostic Management

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

Name	*Description*	*ID*
SEV_UDS_SECURITY_ACCESS_NEEDED	Tester has sent a diagnostic request without meeting the server's security level requirements for that service. NRC 0x33 (securityAccessDenied) was returned.	100

▽

△

Name	Description	ID
SEV_UDS_AUTHENTICATION_NEEDED	A diagnostic request was received while the required authentication to execute this service is not given. NRC 0x34 (authenticationRequired) was returned.	101
SEV_UDS_SECURITY_ACCESS_SUCCESSFUL	Successful unlocked the ECU (via Security Access SID 0x27)	102
SEV_UDS_SECURITY_ACCESS_FAILED	Unlocking of the ECU (via Security Access SID 0x27) failed	103
SEV_UDS_AUTHENTICATION_SUCCESSFUL	Succesfully authenticated (via Authentication SID 0x29)	104
SEV_UDS_AUTHENTICATION_FAILED	Authentication (via Authentication SID 0x29) failed	105
SEV_UDS_WRITE_DATA_SUCCESSFUL	Diagnostic data identifier has been written by SID 0x2E WriteDataByIdentifier	106
SEV_UDS_WRITE_DATA_FAILED	Change of Diagnostic data identifier has been requested by SID 0x2E WriteDataByIdentifier, but failed	107
SEV_UDS_REQUEST_UP_DOWNLOAD_SUCCESSFUL	An upload / download sequence has been requested successfully with SID 0x34 or SID 0x35	110
SEV_UDS_REQUEST_UP_DOWNLOAD_FAILED	An upload / download sequence has been requested with SID 0x34 or SID 0x35, but failed	111
SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL	A file transfer sequence has been requested successfully with SID 0x38.	112
SEV_UDS_REQUEST_FILE_TRANSFER_FAILED	A file transfer sequence has been requested with SID 0x38, but failed	113
SEV_UDS_COMMUNICATION_CONTROL_SUCCESSFUL	The control of a communication has been requested by service SID 0x28 CommunicationControl successfully.	114
SEV_UDS_COMMUNICATION_CONTROL_FAILED	The control of a communication has been requested by service SID 0x28 CommunicationControl, but failed.	115
SEV_UDS_CLEAR_DTC_SUCCESSFUL	DTC information has been cleared by SID 0x14 Clear DiagnosticInformation.	116
SEV_UDS_CLEAR_DTC_FAILED	Clearing DTC information has been requested by SID 0x14 ClearDiagnosticInformation, but failed.	117
SEV_UDS_CONTROL_DTC_SETTING_SUCCESSFUL	The control of a DTC setting has been requested by service SID 0x85 ControlDTCSetting successfully.	118
SEV_UDS_CONTROL_DTC_SETTING_FAILED	Control of DTC setting has been requested by service SID 0x85 ControlDTCSetting, but failed.	119
SEV_UDS_ECU_RESET_SUCCESSFUL	ECU has been reset by SID 0x11 ECUReset.	120
SEV_UDS_ECU_RESET_FAILED	ECU Reset has been requested by SID 0x11 ECUReset, but failed.	121
SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL	The control of a routine has been requested by service SID 0x31 RoutineControl successfully.	122
SEV_UDS_ROUTINE_CONTROL_FAILED	The control of a routine has been requested by service SID 0x31 RoutineControl, but failed.	123

⌋

[SWS_DM_02015] Reporting security access denied to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service request is not allowed in the current security level, which results in a negative response with NRC 0x33 (securityAccessDenied), the DM shall report a security event SEV_UDS_SECURITY_ACCESS_NEEDED to IdsM (see [13] and table [SWS_DM_02016]) with the context data given in [SWS_DM_02016].⌋

[SWS_DM_02016] Security event context data definition: SEV_UDS_SECURITY_ACCESS_NEEDED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_SECURITY_ACCESS_NEEDED***	
ID	100	
Description	Tester has sent a diagnostic request without meeting the server's security level requirements for that service. NRC 0x33 (securityAccessDenied) was returned.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SID	uint8	
Subfunction	uint8	255: is filled in case the service is without Subfunction
DataIdentifier	uint16	65535: is filled in case the service is without DID
RoutineIdentifier	uint16	65535: is filled in case the service is without RID
ClientSourceAddress	uint16	

⌋

[SWS_DM_02017] Reporting authentication required to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service request is not having the required authentication which results in a negative response with NRC 0x34 (authenticationRequired), the DM shall report a security event SEV_UDS_AUTHENTICATION_NEEDED to IdsM (see [13] and table [SWS_DM_02018]) with the context data given in [SWS_DM_02018].⌋

[SWS_DM_02018] Security event context data definition: SEV_UDS_AUTHENTICATION_NEEDED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_AUTHENTICATION_NEEDED***	
ID	101	
Description	A diagnostic request was received while the required authentication to execute this service is not given. NRC 0x34 (authenticationRequired) was returned.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SID	uint8	
Subfunction	uint8	255: is filled in case the service is without Subfunction
DataIdentifier	uint16	65535: is filled in case the service is without DID
RoutineIdentifier	uint16	65535: is filled in case the service is without RID
ClientSourceAddress	uint16	

⌋

[SWS_DM_02019] Reporting successful security access to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service SecurityAccess with subfunction CompareKey is received, which successfully unlocks the requested security access type, The DM shall report a security event SEV_UDS_SECURITY_ACCESS_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02020]) with the context data given in [SWS_DM_02020].⌋

[SWS_DM_02020] Security event context data definition: SEV_UDS_SECURITY_ACCESS_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_SECURITY_ACCESS_SUCCESSFUL***	
ID	102	
Description	Successful unlocked the ECU (via Security Access SID 0x27)	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02021] Reporting failed security access to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service SecurityAccess is received, which results in a negative response,The DM shall report a security event SEV_UDS_SECURITY_ACCESS_FAILED to IdsM (see [13] and table [SWS_DM_02022]) with the context data given in [SWS_DM_02022].⌋

[SWS_DM_02022] Security event context data definition: SEV_UDS_SECURITY_ACCESS_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_SECURITY_ACCESS_FAILED***	
ID	103	
Description	Unlocking of the ECU (via Security Access SID 0x27) failed	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	

▽

△

SEV Name	SEV_UDS_SECURITY_ACCESS_FAILED	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02023] Reporting successful authentication to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service Authentication with subfunction proofOfOwnership is received, which is successful, The DM shall report a security event SEV_UDS_AUTHENTICATION_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02024]) with the context data given in [SWS_DM_02024].⌋

[SWS_DM_02024] Security event context data definition: SEV_UDS_AUTHENTICATION_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	SEV_UDS_AUTHENTICATION_SUCCESSFUL	
ID	104	
Description	Succesfully authenticated (via Authentication SID 0x29)	
Context Data Version	1	
Context Data	**Data Type**	**Allowed Values**
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02025] Reporting failed authentication to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service Authentication is received, which results in a negative response, The DM shall report a security event SEV_UDS_AUTHENTICATION_FAILED to IdsM (see [13] and table [SWS_DM_02026]) with the context data given in [SWS_DM_02026].⌋

[SWS_DM_02026] Security event context data definition: SEV_UDS_AUTHENTICATION_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_AUTHENTICATION_FAILED***	
ID	105	
Description	Authentication (via Authentication SID 0x29) failed	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02027] Reporting successful writing of data to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service WriteDataByIdentifier is received, which results in a positive response, The DM shall report a security event SEV_UDS_WRITE_DATA_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02028]) with the context data given in [SWS_DM_02028].⌋

[SWS_DM_02028] Security event context data definition: SEV_UDS_WRITE_DATA_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_WRITE_DATA_SUCCESSFUL***	
ID	106	
Description	Diagnostic data identifier has been written by SID 0x2E WriteDataByIdentifier	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
DID	uint16	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02029] Reporting writing of data failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service WriteDataByIdentifier is received, which results in a negative response, The DM shall report a security event SEV_UDS_WRITE_DATA_FAILED to IdsM (see [13] and table [SWS_DM_02030]) with the context data given in [SWS_DM_02030].⌋

[SWS_DM_02030] Security event context data definition: SEV_UDS_WRITE_DATA_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_WRITE_DATA_FAILED***	
ID	107	
Description	Change of Diagnostic data identifier has been requested by SID 0x2E WriteDataByIdentifier, but failed	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
DID	uint16	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02031] Reporting successful up download to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RequestDownload or RequestUpload is received, which results in a positive response, The DM shall report a security event SEV_UDS_REQUEST_UP_DOWNLOAD_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02032]) with the context data given in [SWS_DM_02032].⌋

[SWS_DM_02032] Security event context data definition: SEV_UDS_REQUEST_UP_DOWNLOAD_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_REQUEST_UP_DOWNLOAD_SUCCESSFUL***	
ID	110	
Description	An upload / download sequence has been requested successfully with SID 0x34 or SID 0x35	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SID	uint8	
MemoryAddress	uint32	
MemorySize	uint32	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02033] Reporting up download failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RequestDownload or RequestUpload is received, which results in a negative response, The DM shall report a security event SEV_UDS_REQUEST_UP_DOWNLOAD_FAILED to IdsM (see [13] and table [SWS_DM_02034]) with the context data given in [SWS_DM_02034].⌋

[SWS_DM_02034] Security event context data definition: SEV_UDS_REQUEST_UP_DOWNLOAD_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_REQUEST_UP_DOWNLOAD_FAILED***	
ID	111	
Description	An upload / download sequence has been requested with SID 0x34 or SID 0x35, but failed	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
SID	uint8	
MemoryAddress	uint32	
MemorySize	uint32	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02035] Reporting successful file transfer to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RequestFileTransfer is received, which results in a positive response, The DM shall report a security event SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02036]) with the context data given in [SWS_DM_02036].⌋

[SWS_DM_02036] Security event context data definition: SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL***
ID	112
Description	A file transfer sequence has been requested successfully with SID 0x38.
Context Data Version	1

▽

△

SEV Name	*SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL*	
Context Data	***Data Type***	***Allowed Values***
ModeOfOperation	uint8	AddFile (0x01) DeleteFile (0x02) ReplaceFile (0x03) ReadFile (0x04) ReadDir (0x05) ResumeFile (0x06)
FilePathAndName	uint8 [50]	Each byte of this parameter is encoded in ASCII format.
ClientSourceAddress	uint16	

⌋

[SWS_DM_02037] Reporting file transfer failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RequestFileTransfer is received, which results in a negative response, The DM shall report a security event SEV_UDS_REQUEST_FILE_TRANSFER_FAILED to IdsM (see [13] and table [SWS_DM_02038]) with the context data given in [SWS_DM_02038].⌋

[SWS_DM_02038] Security event context data definition: SEV_UDS_REQUEST_FILE_TRANSFER_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_UDS_REQUEST_FILE_TRANSFER_FAILED*	
ID	113	
Description	A file transfer sequence has been requested with SID 0x38, but failed	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
ModeOfOperation	uint8	AddFile (0x01) DeleteFile (0x02) ReplaceFile (0x03) ReadFile (0x04) ReadDir (0x05) ResumeFile (0x06)
FilePathAndName	uint8 [50]	Each byte of this parameter is encoded in ASCII format.
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02039] Reporting successful communication control to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service CommunicationControl is received, which results in a positive response, The DM shall report a security event

SEV_UDS_COMMUNICATION_CONTROL_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02040]) with the context data given in [SWS_DM_02040].⌋

[SWS_DM_02040] Security event context data definition: SEV_UDS_COMMUNICATION_CONTROL_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_COMMUNICATION_CONTROL_SUCCESSFUL***	
ID	114	
Description	The control of a communication has been requested by service SID 0x28 Communication Control successfully.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02041] Reporting communication control failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service CommunicationControl is received, which results in a negative response, The DM shall report a security event SEV_UDS_COMMUNICATION_CONTROL_FAILED to IdsM (see [13] and table [SWS_DM_02042]) with the context data given in [SWS_DM_02042].⌋

[SWS_DM_02042] Security event context data definition: SEV_UDS_COMMUNICATION_CONTROL_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_COMMUNICATION_CONTROL_FAILED***	
ID	115	
Description	The control of a communication has been requested by service SID 0x28 Communication Control, but failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02043] Reporting successful clearing of dtc to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ClearDiagnosticInformation is received, which results in a positive response, The DM shall report a security event SEV_UDS_CLEAR_DTC_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02044]) with the context data given in [SWS_DM_02044].⌋

[SWS_DM_02044] Security event context data definition: SEV_UDS_CLEAR_DTC_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_CLEAR_DTC_SUCCESSFUL***	
ID	116	
Description	DTC information has been cleared by SID 0x14 ClearDiagnosticInformation.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GroupOfDTC	uint8 [3]	given in the format: HighByte, MiddleByte, LowByte
MemorySelection	uint16	0x0001: PrimaryMemory 0x01XX: XX is the address of the UserDefinedMemory
ClientSourceAddress	uint16	

⌋

[SWS_DM_02045] Reporting clearing of dtc failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ClearDiagnosticInformation is received, which results in a negative response, The DM shall report a security event SEV_UDS_CLEAR_DTC_FAILED to IdsM (see [13] and table [SWS_DM_02046]) with the context data given in [SWS_DM_02046].⌋

[SWS_DM_02046] Security event context data definition: SEV_UDS_CLEAR_DTC_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_CLEAR_DTC_FAILED***	
ID	117	
Description	Clearing DTC information has been requested by SID 0x14 ClearDiagnosticInformation, but failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***

▽

△

SEV Name	*SEV_UDS_CLEAR_DTC_FAILED*	
GroupOfDTC	uint8 [3]	given in the format: HighByte, MiddleByte, LowByte
MemorySelection	uint16	0x0001: PrimaryMemory 0x01XX: XX is the address of the UserDefinedMemory
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02047] Reporting successful control of dtc to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ControlDTCSetting is received, which results in a positive response, The DM shall report a security event SEV_UDS_CONTROL_DTC_SETTING_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02048]) with the context data given in [SWS_DM_02048].⌋

[SWS_DM_02048] Security event context data definition: SEV_UDS_CONTROL_DTC_SETTING_SUCCESSFUL

Status: DRAFT

Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_UDS_CONTROL_DTC_SETTING_SUCCESSFUL*	
ID	118	
Description	The control of a DTC setting has been requested by service SID 0x85 ControlDTCSetting successfully.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02049] Reporting control of dtc failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ControlDTCSetting is received, which results in a negative response, The DM shall report a security event SEV_UDS_CONTROL_DTC_SETTING_FAILED to IdsM (see [13] and table [SWS_DM_02050]) with the context data given in [SWS_DM_02050].⌋

[SWS_DM_02050] Security event context data definition: SEV_UDS_CONTROL_DTC_SETTING_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_CONTROL_DTC_SETTING_FAILED***	
ID	119	
Description	Control of DTC setting has been requested by service SID 0x85 ControlDTCSetting, but failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02051] Reporting successful reset of ECU to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ECUReset is received, which results in a positive response, The DM shall report a security event SEV_UDS_ECU_RESET_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02052]) with the context data given in [SWS_DM_02052].

Hint: the report of the security event is done after RequestReset() and before calling ExecuteReset()⌋

[SWS_DM_02052] Security event context data definition: SEV_UDS_ECU_RESET_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_ECU_RESET_SUCCESSFUL***	
ID	120	
Description	ECU has been reset by SID 0x11 ECUReset.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02053] Reporting reset of ECU failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service ECUReset is received, which results in a negative response, The DM shall report a security event SEV_UDS_ECU_RESET_FAILED to IdsM (see [13] and table [SWS_DM_02054]) with the context data given in [SWS_DM_02054].⌋

[SWS_DM_02054] Security event context data definition: SEV_UDS_ECU_RESET_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_ECU_RESET_FAILED***	
ID	121	
Description	ECU Reset has been requested by SID 0x11 ECUReset, but failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Subfunction	uint8	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

[SWS_DM_02055] Reporting successful routine control to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RoutineControl is received, which results in a positive response, The DM shall report a security event SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL to IdsM (see [13] and table [SWS_DM_02056]) with the context data given in [SWS_DM_02056].⌋

[SWS_DM_02056] Security event context data definition: SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL***	
ID	122	
Description	The control of a routine has been requested by service SID 0x31 RoutineControl successfully.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***

▽

△

SEV Name	*SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL*	
RID	uint16	
Subfunction	uint8	
ClientSourceAddress	uint16	

⌋

[SWS_DM_02057] Reporting routine control failed to IdsM

Upstream requirements: RS_Ids_00810

⌈In case a diagnostic service RoutineControl is received, which results in a negative response, The DM shall report a security event SEV_UDS_ROUTINE_CONTROL_FAILED to IdsM (see [13] and table [SWS_DM_02058]) with the context data given in [SWS_DM_02058].⌋

[SWS_DM_02058] Security event context data definition: SEV_UDS_ROUTINE_CONTROL_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_UDS_ROUTINE_CONTROL_FAILED*	
ID	123	
Description	The control of a routine has been requested by service SID 0x31 RoutineControl, but failed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
RID	uint16	
Subfunction	uint8	
ClientSourceAddress	uint16	
NegativeResponseCode	uint8	

⌋

7.5.2 Log Messages

This functional cluster does not define any non-verbose log messages (i.e., modelled DLT messages).

7.5.3 Violation Messages

This section lists all violation messages (i.e., DLT messages logged for Violations according to [SWS_CORE_00021]) defined by this functional cluster.

Dlt-Message	InstanceSpecifierMappingIntegrityViolation		
Description	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {processIdentifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"		
MessageId	0x80001ffc		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Identifier of the process that caused the violation.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	NoUnit
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	NoUnit
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	NoUnit

Dlt-Message	PortInterfaceMappingViolation		
Description	The type of mapping does not match the expected type of PortInterface: {portInterfaceTypeName} referenced by a {mappingTypeName}. String format: "Violation detected in {processIdentifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"		
MessageId	0x80001ffb		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Identifier of the process that caused the violation.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	NoUnit
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	NoUnit
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	NoUnit

Dlt-Message	ProcessMappingViolation		
Description	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"		
MessageId	0x80001ffa		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Identifier of the process that caused the violation.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	NoUnit
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	NoUnit
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	NoUnit

Dlt-Message	InstanceSpecifierAlreadyInUseViolation		
Description	Violation message that is sent in case a constructor in the ara framework was called with an Instance Specifier already in use in this process. String format: "Violation detected in {processIdentifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"		
MessageId	0x80001ff9		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Identifier of the process that caused the violation.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	NoUnit
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	NoUnit
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	NoUnit

[SWS_DM_02061] ViolationMessage InvalidDebouncingAlgorithmViolation

Status: DRAFT

Upstream requirements: RS_Diag_04068, RS_AP_00149

⌈

Dlt-Message	InvalidDebouncingAlgorithmViolation		
Description	Sent in case of a mismatch between the debouncing parameters in DEXT and the algorithm provided during object creation. String format: "Violation detected in {processIdentifier} at {location} where the debouncing algorithm between DEXT: {modelledAlgorithm} and runtime: {providedAlogrithm} does not match.".		
MessageId	0x80003fff		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Meta-model identifier of the process that caused the violation, i.e. short name path with '/' as a separator.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example \\\{filename}:\\\{linenumber}.	uint8 [encoding UTF-8]	NoUnit
modelled Algorithm	Name of the mapped debounce algorithm according to DiagnosticEventToDebounceAlgorithmMapping	uint8 [encoding UTF-8]	NoUnit
provided Alogrithm	Name of the algorithm provided during object construction	uint8 [encoding UTF-8]	NoUnit

⌋

[SWS_DM_02062] ViolationMessage UnexpectedMonitorActionHandlingViolation

Status: DRAFT
Upstream requirements: RS_Diag_04179, RS_Diag_04179

⌈

Dlt-Message	UnexpectedMonitorActionHandlingViolation		
Description	Sent in case ara::diag::Monitor::ReportMonitorAction is called with a parameter "action" value that is not supported by the used debouncing method.		
MessageId	0x80003ffe		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Meta-model identifier of the process that caused the violation, i.e. short name path with '/' as a separator.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example \\\{filename}:\\\{linenumber}.	uint8 [encoding UTF-8]	NoUnit
monitorAction	Value of the requested "action" parameter of ara::diag::Monitor::ReportMonitorAction	uint8 [encoding UTF-8]	NoUnit
debouncing Method	Value of the used debouncing method	uint8 [encoding UTF-8]	NoUnit

⌋

7.5.4 Production Errors

This functional cluster does not define any production errors (i.e., Diagnostic Events).

8 API specification

This chapter provides a reference of the APIs defined by this functional cluster. The API is described in the following chapters in tables. Table 8.1 explains the content that is described in such an API table.

Kind:	Defines the kind of the declaration that this API table describes. The following values are supported: • class (Declaration of a class) • function (Declaration of a member or non-member function) • struct (Declaration of a structure) • type alias (Declaration of a type alias) • enumeration (Declaration of an enumeration) • variable (Declaration of a variable)	
Header File:	Defines the header file to be included according to [SWS_CORE_90001]	
Forwarding Header File:	Defines the forwarding header file to be included according to [SWS_CORE_90001]	
Scope:	Defines the scope that may be a namespace (in case of a class or non-member function) or a class declaration (in case of a member)	
Symbol:	Entity name	
Thread Safety:	Defines whether a function is thread-safe, not thread-safe, or conditional according to [SWS_CORE_13200] and [SWS_CORE_13202]	
Syntax:	Description of C++ syntax	
Template Param:	Template parameter (0..*)	Template parameter(s) used to parametrize the template
Parameters (in):	Parameter declaration (0..*)	Parameter(s) that are passed to the function
Parameters (out):	Parameter declaration (0..*)	Parameter(s) that are returned to the caller
Return Value:	Return type	Type of the value that the function returns
Exception Safety:	Defines whether a function is exception-safe, not exception safe or conditionally exception safe	
Exceptions:	List of exceptions that may be thrown from the function	
Violations:	List of violations that may occur in the function	
Errors:	Error type (0..*)	List of defined error codes that may be returned by the function with their recoverability class defined in [RS_AP_00160]. APIs can be extended with vendor-specific error codes. These are not part of the AUTOSAR SWS specifications
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

This chapter lists all provided and required C++ API interfaces of the DM. The C++ API interfaces are divided into two parts:

- Diagnostic Application interface
  A DiagnosticPortInterfaces is representing a corresponding code instance. The deployment is simplified due to a direct mapping to DiagnosticObject in DEXT.

8.1 Diagnostic Communication-related APIs

8.2 Header: ara/diag/authentication.h

8.2.1 Class: Authentication

[SWS_DM_01123] Definition of API class ara::diag::Authentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/authentication.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Authentication
Syntax:	`class Authentication {...};`
Description:	Class to implement the Service Authentication interfaces to application.

⌋

8.2.1.1 Public Member Functions

8.2.1.1.1 Special Member Functions

8.2.1.1.1.1 Move Constructor

[SWS_DM_01610] Definition of API function ara::diag::Authentication::Authentication

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`Authentication (Authentication &&) noexcept=delete;`
Description:	Move constructor of Authentication.

⌋

8.2.1.1.1.2 Move Assignment Operator

[SWS_DM_01608] Definition of API function ara::diag::Authentication::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`Authentication & operator= (Authentication &&)=delete;`
Description:	Move assignment operator of Authentication.

⌋

8.2.1.1.1.3 Destructor

[SWS_DM_01125] Definition of API function ara::diag::Authentication::~Authentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`virtual ~Authentication () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of Authentication .

⌋

8.2.1.1.2 Constructors

8.2.1.1.2.1 Authentication

[SWS_DM_01124] Definition of API function ara::diag::Authentication::Authentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/authentication.h"	
Scope:	`class ara::diag::Authentication`	
Syntax:	`explicit Authentication (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticAuthentication service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticAuthenticationPortMapping` needs to be typed by a `DiagnosticAuthenticationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of Authentication.	

⌋

8.2.1.1.2.2 Authentication

[SWS_DM_01609] Definition of API function ara::diag::Authentication::Authentication

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`Authentication (Authentication &)=delete;`
Description:	Authentication shall be a single not copy-able instance.

⌋

8.2.1.1.3 Member Functions

8.2.1.1.3.1 Offer

[SWS_DM_01130] Definition of API function ara::diag::Authentication::Offer

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/authentication.h"	
Scope:	`class ara::diag::Authentication`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.2.1.1.3.2 StopOffer

[SWS_DM_01131] Definition of API function ara::diag::Authentication::StopOffer

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.2.1.1.3.3 VerifyCertificateBidirectional

[SWS_DM_01127] Definition of API function ara::diag::Authentication::VerifyCertificateBidirectional

Status: DRAFT
Upstream requirements: RS_Diag_04251, RS_AP_00128

⌈

Kind:	function	
Header file:	#include "ara/diag/authentication.h"	
Scope:	`class ara::diag::Authentication`	
Syntax:	`virtual ara::core::Future< std::tuple< ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte > > > VerifyCertificateBidirectional (ara::core::Byte communicationConfiguration, ara::core::Span< const ara::core::Byte > clientCertificate, ara::core::Span< const ara::core::Byte > client Challenge, const MetaInfo &metaInfo, CancellationHandler cancellation Handler) noexcept=0;`	
Parameters (in):	communication Configuration	As defined in ISO14229-1:2020, this parameter provides information about how to proceed with security in further diagnostic communication after the Authentication.
	clientCertificate	The certificate that is received from the tester during Bidirectional Authentication, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	clientChallenge	As defined in ISO14229-1:2020, this parameter provides the challenge received from the tester during Bidirectional Authentication, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.

▽

△

	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< std::tuple< ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte > > >	Challenge created by the application, Certificate of the server, ProofOfOwnership calculated by the server, Ephemeral Public Key of server
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	This function accepts the certificate and challenge received from the tester, verifies the certificate, and creates a challenge, Ephemeral Public Key and Proof Of Ownership that must be returned to the tester. The function also returns the server certificate that will be used by the tester to verify the Proof Of Ownership. This callback may be called re-entrant to requests from different clients	

⌋

8.2.1.1.3.4 VerifyCertificateUnidirectional

[SWS_DM_01126] Definition of API function ara::diag::Authentication::VerifyCertificateUnidirectional

Status: DRAFT
Upstream requirements: RS_Diag_04251, RS_AP_00128

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`virtual ara::core::Future< std::tuple< ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte > > > VerifyCertificateUnidirectional (ara::core::Byte communicationConfiguration, ara::core::Span< const ara::core::Byte > clientCertificate, ara::core::Span< const ara::core::Byte > clientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`

▽

△

Parameters (in):	communication Configuration	As defined in ISO14229-1:2020, this parameter provides information about how to proceed with security in further diagnostic communication after the Authentication.
	clientCertificate	The certificate that is received from the tester during Unidirectional Authentication, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	clientChallenge	As defined in ISO14229-1:2020, this parameter provides the challenge received fom the tester during Unidirectional Authentication. This parameter has a dependency on the CommunicationConfiguration used, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< std::tuple< ara::core::Vector< ara::core::Byte >, ara::core::Vector< ara::core::Byte > > >	Challenge created by the application, Ephemeral Public Key of Server
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	This function accepts the certificate received from the tester, verifies it, and creates a challenge and ephemeral public key that must be returned to the tester. This callback may be called re-entrant to requests from different clients	

⌋

8.2.1.1.3.5 VerifyOwnership

[SWS_DM_01128] Definition of API function ara::diag::Authentication::VerifyOwnership

Status: DRAFT
Upstream requirements: RS_Diag_04251, RS_AP_00128

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`

▽

△

Syntax:	`virtual ara::core::Future< ara::core::Vector< ara::core::Byte > > VerifyOwnership (ara::core::Span< const ara::core::Byte > clientPOWN, ara::core::Span< const ara::core::Byte > clientEphemeralPublicKey, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	clientPOWN	The Proof Of Ownership provided by the Tester to the previously exchanged Server Challenge, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	clientEphemeralPublic Key	As defined in ISO14229-1:2020, this is the Ephemeral public key generated by the client for Diffie-Hellman key agreement, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >	Session Key Info or error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	This function accepts the Proof Of Ownership received from the tester and verifies it with the Public Key of the certificate received in the verifycertificateunidirectional/ verifycertificatebidirectional against the server challenge created in the last call to verifycertificateunidirectional/ verifycertificatebidirectional.	

⌋

8.2.1.1.3.6 operator=

[SWS_DM_01607] Definition of API function ara::diag::Authentication::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	`class ara::diag::Authentication`
Syntax:	`Authentication & operator= (Authentication &)=delete;`
Description:	Authentication shall be a single not assignable instance.

⌋

8.3 Header: ara/diag/cancellation_handler.h

8.3.1 Class: CancellationHandler

[SWS_DM_00608] Definition of API class ara::diag::CancellationHandler

Upstream requirements: RS_Diag_04169

⌈

Kind:	class
Header file:	#include "ara/diag/cancellation_handler.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	CancellationHandler
Syntax:	`class CancellationHandler final {...};`
Description:	CancellationHandler contains a shared state if the processing should be canceled .

⌋

8.3.1.1 Public Member Types

8.3.1.1.1 Type Alias: CancellationHandlerSetNotifier

[SWS_DM_02075] Definition of API type ara::diag::CancellationHandler::CancellationHandlerSetNotifier

Upstream requirements: RS_Diag_04169

⌈

Kind:	type alias
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	`class ara::diag::CancellationHandler`
Symbol:	CancellationHandlerSetNotifier
Syntax:	`using CancellationHandlerSetNotifier = std::function<void(void)>;`
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the diagnostic service execution is canceled in DM. .

⌋

8.3.1.2 Public Member Functions

8.3.1.2.1 Special Member Functions

8.3.1.2.1.1 Copy Constructor

[SWS_DM_00611] Definition of API function ara::diag::CancellationHandler::CancellationHandler

Upstream requirements: RS_Diag_04169, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	`class ara::diag::CancellationHandler`
Syntax:	`CancellationHandler (const CancellationHandler &)=delete;`
Description:	CancellationHandler shall be a single not copy-able instance.

⌋

8.3.1.2.1.2 Default Constructor

[SWS_DM_00609] Definition of API function ara::diag::CancellationHandler::CancellationHandler

Upstream requirements: RS_Diag_04169, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	`class ara::diag::CancellationHandler`
Syntax:	`CancellationHandler ()=delete;`
Description:	CancellationHandler shall not be constructable by applications but the DM core only.

⌋

8.3.1.2.1.3 Move Constructor

[SWS_DM_00610] Definition of API function ara::diag::CancellationHandler::CancellationHandler

Upstream requirements: RS_AP_00133, RS_Diag_04169, RS_AP_00145

⌈

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	`class ara::diag::CancellationHandler`	
Syntax:	`CancellationHandler (CancellationHandler &&) noexcept=default;`	
DIRECTION NOT DEFINED	CancellationHandler &&	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of CancellationHandler.	

⌋

8.3.1.2.1.4 Copy Assignment Operator

[SWS_DM_00612] Definition of API function ara::diag::CancellationHandler::operator=

Upstream requirements: RS_AP_00133, RS_Diag_04169, RS_AP_00145

⌈

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	`class ara::diag::CancellationHandler`	
Syntax:	`CancellationHandler & operator= (const CancellationHandler &) noexcept=default;`	
DIRECTION NOT DEFINED	const Cancellation Handler &	--
Return value:	CancellationHandler &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of CancellationHandler .	

⌋

8.3.1.2.2 Member Functions

8.3.1.2.2.1 IsCanceled

[SWS_DM_00614] Definition of API function ara::diag::CancellationHandler::IsCanceled

Upstream requirements: RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	`class ara::diag::CancellationHandler`	
Syntax:	`bool IsCanceled () const noexcept;`	
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Returns true in if the diagnostic service execution is cancelled in DM. .	

⌋

8.3.1.2.2.2 SetNotifier

[SWS_DM_00615] Definition of API function ara::diag::CancellationHandler::SetNotifier

Upstream requirements: RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	`class ara::diag::CancellationHandler`	
Syntax:	`void SetNotifier (CancellationHandlerSetNotifier notifier) noexcept;`	
Parameters (in):	notifier	Notification function that is called upon diagnostic service execution is canceled in DM.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Registering a notifier function which is called if the diagnostic service execution is canceled in DM. A consecutive call of this method will overwrite the previous registered notifier. .	

⌋

8.3.1.2.2.3 operator=

[SWS_DM_00613] Definition of API function ara::diag::CancellationHandler::operator=

Upstream requirements: RS_Diag_04169, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	`class ara::diag::CancellationHandler`
Syntax:	`CancellationHandler & operator= (CancellationHandler &)=delete;`
Description:	CancellationHandler shall be a single not assignable instance.

⌋

8.4 Header: ara/diag/client_authentication.h

8.4.1 Class: ClientAuthentication

[SWS_DM_01132] Definition of API class ara::diag::ClientAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/client_authentication.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	ClientAuthentication
Syntax:	`class ClientAuthentication final {...};`
Description:	Interface for the application to inform the Diagnostic Server instance about the authentication states and the user roles that are currently authenticated.

⌋

8.4.1.1 Public Member Types

8.4.1.1.1 Type Alias: ClientAuthenticationSetNotifier

[SWS_DM_02077] Definition of API type ara::diag::ClientAuthentication::ClientAuthenticationSetNotifier

Upstream requirements: RS_Diag_04251

⌈

Kind:	type alias
Header file:	#include "ara/diag/client_authentication.h"
Scope:	`class ara::diag::ClientAuthentication`
Symbol:	ClientAuthenticationSetNotifier
Syntax:	`using ClientAuthenticationSetNotifier = std::function<void(DiagnosticAuthState)>;`
Thread Safety:	not thread-safe
Description:	Notifier Function that is called by the Diagnostic Server instance when an Authentication Status Change Occurs. This may be used, for. E.g, to notify the application when a transition to kDe Authenticated State occurred due to an S3 timeout. .

⌋

8.4.1.1.2 Type Alias: DiagnosticAuthRole

[SWS_DM_01134] Definition of API type ara::diag::ClientAuthentication::DiagnosticAuthRole

Upstream requirements: RS_Diag_04251

⌈

Kind:	type alias
Header file:	#include "ara/diag/client_authentication.h"
Scope:	`class ara::diag::ClientAuthentication`
Symbol:	DiagnosticAuthRole
Syntax:	`using DiagnosticAuthRole = ara::core::String;`
Description:	The Supported values for the Diagnostic Authentication roles are specified in the Diagnostic Extract.

⌋

8.4.1.1.3 Enumeration: DiagnosticAuthState

[SWS_DM_01133] Definition of API enum ara::diag::ClientAuthentication::DiagnosticAuthState

Upstream requirements: RS_Diag_04251

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/client_authentication.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Symbol:	DiagnosticAuthState	
Underlying type:	std::uint8_t	
Syntax:	`enum class DiagnosticAuthState : std::uint8_t {...};`	
Values:	kDeAuthenticated= 0x00	No Diagnostic Clients are currently authenticated.
	kAuthenticated= 0x01	A Diagnostic Client is currently authen ticated.
Description:	Possible values of the Authentication State of the client.	

⌋

8.4.1.2 Public Member Functions

8.4.1.2.1 Special Member Functions

8.4.1.2.1.1 Move Constructor

[SWS_DM_01137] Definition of API function ara::diag::ClientAuthentication::ClientAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`ClientAuthentication (ClientAuthentication &&other) noexcept=default;`	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of ClientAuthentication.	

⌋

8.4.1.2.1.2 Copy Constructor

[SWS_DM_01139] Definition of API function ara::diag::ClientAuthentication::ClientAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication.h"
Scope:	`class ara::diag::ClientAuthentication`
Syntax:	`ClientAuthentication (ClientAuthentication const &other)=delete;`
Description:	Copy constructor of ClientAuthentication cannot be used.

⌋

8.4.1.2.1.3 Destructor

[SWS_DM_01136] Definition of API function ara::diag::ClientAuthentication::~ClientAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication.h"
Scope:	`class ara::diag::ClientAuthentication`
Syntax:	`~ClientAuthentication () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ClientAuthentication .

⌋

8.4.1.2.2 Member Functions

8.4.1.2.2.1 Authenticate

[SWS_DM_01142] Definition of API function ara::diag::ClientAuthentication::Authenticate

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`ara::core::Result< ClientAuthenticationHandle > Authenticate (ara::core::Vector< DiagnosticAuthRole > userRoles) noexcept;`	
Parameters (in):	userRoles	The user roles to set on the diagnostic client
Return value:	ara::core::Result< ClientAuthenticationHandle >	A handler of the Authentication State, which can be used by the application to set or extend the DynamicAccessList
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to report the authenticated state to the Diagnostic Server instance. The authentication could be either done using the Authentication interfaces of the Diagnostic Server instance, or through other means in the application.	

⌋

8.4.1.2.2.2 GetState

[SWS_DM_01143] Definition of API function ara::diag::ClientAuthentication::GetState

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`ara::core::Result< DiagnosticAuthState > GetState () const noexcept;`	
Return value:	ara::core::Result< DiagnosticAuthState >	The Authentication State of the Diagnostic Client or error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to query the current authentication state of the diagnostic client.	

⌋

8.4.1.2.2.3 OverrideDefaultRoles

[SWS_DM_01141] Definition of API function ara::diag::ClientAuthentication::OverrideDefaultRoles

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`ara::core::Result< ClientAuthenticationHandle > OverrideDefaultRoles (ara::core::Vector< DiagnosticAuthRole > defaultRoles, std::chrono::milliseconds timeout) noexcept;`	
Parameters (in):	defaultRoles	The default roles requested by the application, to be set on the diagnostic client
	timeout	The timeout until which the override request is active
Return value:	ara::core::Result< ClientAuthenticationHandle >	Operation result
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This method is used by the application to temporarily change the default AuthenticationRole for a Diagnostic Server Instance. The diagnostic services allowed in the passed defaultRoles are now accessible to the tester for a time period defined in the parameter timeout.	

⌋

8.4.1.2.2.4 SetNotifier

[SWS_DM_01144] Definition of API function ara::diag::ClientAuthentication::SetNotifier

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`ara::core::Result< void > SetNotifier (ClientAuthenticationSetNotifier notifier) noexcept;`	
Parameters (in):	notifier	The notifier to call on state transition
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to set a Notifier Function that shall be called by the Diagnostic Server instance when an Authentication Status Change Occurs. This may be used, for. E.g, to notify the application when a transition to kDeAuthenticated State occurred due to an S3 timeout. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.4.1.2.2.5 operator=

[SWS_DM_01138] Definition of API function ara::diag::ClientAuthentication::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication.h"	
Scope:	`class ara::diag::ClientAuthentication`	
Syntax:	`auto operator= (ClientAuthentication &&other) & noexcept -> ClientAuthentication & =default;`	
Parameters (in):	other	Object to move-assign from.
Return value:	ClientAuthentication & =default	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of ClientAuthentication.	

⌋

8.4.1.2.2.6 operator=

[SWS_DM_01140] Definition of API function ara::diag::ClientAuthentication::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication.h"
Scope:	`class ara::diag::ClientAuthentication`
Syntax:	`auto operator= (ClientAuthentication const &other) -> ClientAuthentication &=delete;`
Description:	Copy assignment operator of ClientAuthentication cannot be used.

⌋

8.5 Header: ara/diag/client_authentication_handle.h

8.5.1 Class: ClientAuthenticationHandle

[SWS_DM_01145] Definition of API class ara::diag::ClientAuthenticationHandle

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/client_authentication_handle.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	ClientAuthenticationHandle
Syntax:	`class ClientAuthenticationHandle final {...};`
Description:	Definition of the ClientAuthenticationHandle which is returned to the application when an AuthenticationState is set by the application.

⌋

8.5.1.1 Public Member Functions

8.5.1.1.1 Special Member Functions

8.5.1.1.1.1 Copy Constructor

[SWS_DM_01150] Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication_handle.h"
Scope:	`class ara::diag::ClientAuthenticationHandle`
Syntax:	`ClientAuthenticationHandle (ClientAuthenticationHandle const &other)=delete;`
Description:	Copy constructor of ClientAuthenticationHandle cannot be used.

⌋

8.5.1.1.1.2 Default Constructor

[SWS_DM_01146] Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication_handle.h"
Scope:	`class ara::diag::ClientAuthenticationHandle`
Syntax:	`ClientAuthenticationHandle () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructor of ClientAuthenticationHandle .

⌋

8.5.1.1.1.3 Move Constructor

[SWS_DM_01148] Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Scope:	`class ara::diag::ClientAuthenticationHandle`	
Syntax:	`ClientAuthenticationHandle (ClientAuthenticationHandle &&other) noexcept=default;`	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of ClientAuthenticationHandle.	

⌋

8.5.1.1.1.4 Destructor

[SWS_DM_01147] Definition of API function ara::diag::ClientAuthenticationHandle::~ClientAuthenticationHandle

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication_handle.h"
Scope:	`class ara::diag::ClientAuthenticationHandle`
Syntax:	`~ClientAuthenticationHandle () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ClientAuthenticationHandle .

⌋

8.5.1.1.2 Member Functions

8.5.1.1.2.1 Append

[SWS_DM_01152] Definition of API function ara::diag::ClientAuthenticationHandle::Append

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Scope:	`class ara::diag::ClientAuthenticationHandle`	
Syntax:	`ara::core::Result< void > Append (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;`	
Parameters (in):	dynamicAccessList	The DynamicAccessList to be appended in the client
Return value:	ara::core::Result< void >	void
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to append a DynamicAccessList to the already existing DynamicAccessList of a Diagnostic Conversation.	

⌋

8.5.1.1.2.2 Refresh

[SWS_DM_01155] Definition of API function ara::diag::ClientAuthenticationHandle::Refresh

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Scope:	`class ara::diag::ClientAuthenticationHandle`	
Syntax:	`ara::core::Result< void > Refresh () noexcept;`	
Return value:	ara::core::Result< void >	void
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics

▽

△

		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to refresh the timer that was started by Authenticate or OverrideDefaultRoles. If both Methods were previously called, both timers are refreshed.	

⌋

8.5.1.1.2.3 Revoke

[SWS_DM_01154] Definition of API function ara::diag::ClientAuthenticationHandle::Revoke

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Scope:	`class ara::diag::ClientAuthenticationHandle`	
Syntax:	`ara::core::Result< void > Revoke () noexcept;`	
Return value:	ara::core::Result< void >	void
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to de-authenticate a client, and also to clear the Dynamic AccessList and any overridden defaults.	

⌋

8.5.1.1.2.4 Set

[SWS_DM_01153] Definition of API function ara::diag::ClientAuthenticationHandle::Set

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication_handle.h"
Scope:	`class ara::diag::ClientAuthenticationHandle`

▽

△

Syntax:	`ara::core::Result< void > Set (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;`	
Parameters (in):	dynamicAccessList	The new DynamicAccessList to be set in the client
Return value:	ara::core::Result< void >	void
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to set/replace a DynamicAccessList of a diagnostic conversation.	

⌋

8.5.1.1.2.5 operator=

[SWS_DM_01149] Definition of API function ara::diag::ClientAuthenticationHandle::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Scope:	`class ara::diag::ClientAuthenticationHandle`	
Syntax:	`auto operator= (ClientAuthenticationHandle &&other) & noexcept -> ClientAuthenticationHandle & =default;`	
Parameters (in):	other	Object to move-assign from.
Return value:	ClientAuthentication Handle & =default	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of ClientAuthenticationHandle.	

⌋

8.5.1.1.2.6 operator=

[SWS_DM_01151] Definition of API function ara::diag::ClientAuthenticationHandle::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/client_authentication_handle.h"
Scope:	`class ara::diag::ClientAuthenticationHandle`
Syntax:	`auto operator= (ClientAuthenticationHandle const &other) -> ClientAuthenticationHandle &=delete;`
Description:	Copy assignment operator of CancellationHandler cannot be used.

⌋

8.6 Header: ara/diag/communication_control.h

8.6.1 Class: CommunicationControl

[SWS_DM_00804] Definition of API class ara::diag::CommunicationControl

Upstream requirements: RS_Diag_04196

⌈

Kind:	class
Header file:	#include "ara/diag/communication_control.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	CommunicationControl
Syntax:	`class CommunicationControl {...};`
Description:	CommunicationControl interface.

⌋

8.6.1.1 Public Member Functions

8.6.1.1.1 Special Member Functions

8.6.1.1.1.1 Move Constructor

[SWS_DM_01678] Definition of API function ara::diag::CommunicationControl::CommunicationControl

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	`class ara::diag::CommunicationControl`
Syntax:	`CommunicationControl (CommunicationControl &&) noexcept=delete;`
Description:	Move constructor of CommunicationControl.

⌋

8.6.1.1.1.2 Move Assignment Operator

[SWS_DM_01676] Definition of API function ara::diag::CommunicationControl::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	`class ara::diag::CommunicationControl`
Syntax:	`CommunicationControl & operator= (CommunicationControl &&)=delete;`
Description:	Move assignment operator of CommunicationControl.

⌋

8.6.1.1.1.3 Destructor

[SWS_DM_00807] Definition of API function ara::diag::CommunicationControl::~CommunicationControl

Upstream requirements: RS_AP_00134, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	`class ara::diag::CommunicationControl`
Syntax:	`virtual ~CommunicationControl () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class CommunicationControl .

⌋

8.6.1.1.2 Constructors

8.6.1.1.2.1 CommunicationControl

[SWS_DM_00806] Definition of API function ara::diag::CommunicationControl::CommunicationControl

Upstream requirements: RS_AP_00137, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	`class ara::diag::CommunicationControl`	
Syntax:	`explicit CommunicationControl (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticComControl Interface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticComControlInterface needs to be referneced by a DiagnosticServiceGenericMapping.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Class for an CommunicationControl.	

⌋

8.6.1.1.2.2 CommunicationControl

[SWS_DM_01677] Definition of API function ara::diag::CommunicationControl::CommunicationControl

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	class ara::diag::CommunicationControl
Syntax:	CommunicationControl (CommunicationControl &)=delete;
Description:	CommunicationControl shall be a single not copy-able instance.

⌋

8.6.1.1.3 Member Functions

8.6.1.1.3.1 CommCtrlRequest

[SWS_DM_00808] Definition of API function ara::diag::CommunicationControl::CommCtrlRequest

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	`class ara::diag::CommunicationControl`	
Syntax:	`virtual ara::core::Future< void > CommCtrlRequest (ComCtrlRequest ParamsType controlType, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	controlType	All UDS request parameters packed into a structure since it holds optional elements
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >	--
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for CommunicationControl (x028) with any subfunction as subfunction value is part of argument list.	

⌋

8.6.1.1.3.2 Offer

[SWS_DM_00809] Definition of API function ara::diag::CommunicationControl::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	`class ara::diag::CommunicationControl`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.6.1.1.3.3 StopOffer

[SWS_DM_00810] Definition of API function ara::diag::CommunicationControl::StopOffer

Upstream requirements: RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	`class ara::diag::CommunicationControl`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.6.1.1.3.4 operator=

[SWS_DM_01675] Definition of API function ara::diag::CommunicationControl::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	`class ara::diag::CommunicationControl`
Syntax:	`CommunicationControl & operator= (CommunicationControl &)=delete;`
Description:	CommunicationControl shall be a single not assignable instance.

⌋

8.6.2 Struct: ComCtrlRequestParamsType

[SWS_DM_00805] Definition of API class ara::diag::CommunicationControl::ComCtrlRequestParamsType

Upstream requirements: RS_Diag_04196

⌈

Kind:	struct
Header file:	#include "ara/diag/communication_control.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::CommunicationControl`
Symbol:	ComCtrlRequestParamsType
Syntax:	`struct ComCtrlRequestParamsType {...};`
Description:	ComCtrlRequestParamsType is a structure, which holds all parameters of an UDS 0x28 communicationControl request.

⌋

8.6.2.1 Public Member Variables

8.6.2.1.1 communicationType

[SWS_DM_01563] Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::communicationType

Upstream requirements: RS_Diag_04196

⌈

Kind:	variable
Header file:	#include "ara/diag/communication_control.h"
Scope:	`struct ara::diag::CommunicationControl::ComCtrlRequestParamsType`
Symbol:	communicationType
Type:	`std::uint8_t`
Syntax:	`std::uint8_t communicationType;`
Description:	CommunicationType from UDS request.

⌋

8.6.2.1.2 controlType

[SWS_DM_01562] Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::controlType

Upstream requirements: RS_Diag_04196

⌈

Kind:	variable
Header file:	#include "ara/diag/communication_control.h"
Scope:	`struct ara::diag::CommunicationControl::ComCtrlRequestParamsType`
Symbol:	controlType
Type:	`std::uint8_t`
Syntax:	`std::uint8_t controlType;`
Description:	ControlType from UDS request.

⌋

8.6.2.1.3 nodeIdentificationNumber

[SWS_DM_01564] Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::nodeIdentificationNumber

Upstream requirements: RS_Diag_04196

⌈

Kind:	variable
Header file:	#include "ara/diag/communication_control.h"
Scope:	`struct ara::diag::CommunicationControl::ComCtrlRequestParamsType`
Symbol:	nodeIdentificationNumber
Type:	`std::uint16_t`
Syntax:	`std::uint16_t nodeIdentificationNumber;`
Description:	Node identifier to which the request is addressed to.

⌋

8.7 Header: ara/diag/concurrency.h

8.7.1 Non-Member Types

8.7.1.1 Enumeration: ConcurrencyType

[SWS_DM_00935] Definition of API enum ara::diag::ConcurrencyType

Upstream requirements: RS_Diag_04166

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/concurrency.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	ConcurrencyType	
Underlying type:	std::uint8_t	
Syntax:	`enum class ConcurrencyType : std::uint8_t {...};`	
Values:	kConcurrent= 0x00	API allows concurrent access (thread-safe)
	kNotConcurrent= 0x01	Concurrent access not allowed (not thread-safe)
Description:	Specifies the Concurrency types.	

⌋

8.7.2 Struct: DataIdentifierConcurrencyType

[SWS_DM_00936] Definition of API class ara::diag::DataIdentifierConcurrency Type

Upstream requirements: RS_Diag_04166

⌈

Kind:	struct
Header file:	#include "ara/diag/concurrency.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataIdentifierConcurrencyType
Syntax:	`struct DataIdentifierConcurrencyType {...};`
Description:	Specifies the Concurrency type of a DataIdentifier related port.

⌋

8.7.2.1 Public Member Variables

8.7.2.1.1 read

[SWS_DM_00937] Definition of API variable ara::diag::DataIdentifierConcurrency Type::read

Upstream requirements: RS_Diag_04166

⌈

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	`struct ara::diag::DataIdentifierConcurrencyType`
Symbol:	read
Type:	`ConcurrencyType`
Syntax:	`ConcurrencyType read;`
Description:	Concurrency type for Reads.

⌋

8.7.2.1.2 readWrite

[SWS_DM_00939] Definition of API variable ara::diag::DataIdentifierConcurrency Type::readWrite

Upstream requirements: RS_Diag_04166

⌈

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	`struct ara::diag::DataIdentifierConcurrencyType`
Symbol:	readWrite
Type:	`ConcurrencyType`
Syntax:	`ConcurrencyType readWrite;`
Description:	Concurrency type for calling Read and Write methods in a concurrent way. If set to `kConcurrent` the DM can call Read and Write APIs concurrently.

⌋

8.7.2.1.3 write

[SWS_DM_00938] Definition of API variable ara::diag::DataIdentifierConcurrency Type::write

Upstream requirements: RS_Diag_04166

⌈

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	`struct ara::diag::DataIdentifierConcurrencyType`
Symbol:	write
Type:	`ConcurrencyType`
Syntax:	`ConcurrencyType write;`
Description:	Concurrency type for Writes.

⌋

8.8 Header: ara/diag/conversation.h

8.8.1 Non-Member Types

8.8.1.1 Enumeration: ActivityStatusType

[SWS_DM_00690] Definition of API enum ara::diag::ActivityStatusType

Upstream requirements: RS_AP_00125, RS_Diag_04166, RS_Diag_04169, RS_Diag_04209

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/conversation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	ActivityStatusType	
Underlying type:	std::uint8_t	
Syntax:	`enum class ActivityStatusType : std::uint8_t {...};`	
Values:	kActive= 0x00	Currently active; i.e. request is currently processed or non-default session is active.
	kInactive= 0x01	Currently not active.
Description:	Type for current activity status.	

⌋

8.8.1.2 Type Alias: SecurityLevelType

[SWS_DM_00705] Definition of API type ara::diag::SecurityLevelType

Upstream requirements: RS_AP_00125

⌈

Kind:	type alias
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	SecurityLevelType
Syntax:	`using SecurityLevelType = std::uint8_t;`
Description:	Type for the active security level. .

⌋

8.8.1.3 Type Alias: SessionControlType

[SWS_DM_00706] Definition of API type ara::diag::SessionControlType

Upstream requirements: RS_AP_00125

⌈

Kind:	type alias
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	SessionControlType
Syntax:	`using SessionControlType = std::uint8_t;`
Description:	Type for the active diagnostic session. .

⌋

8.8.2 Global Variables

8.8.2.1 kDefaultSession

[SWS_DM_01278] Definition of API variable ara::diag::kDefaultSession

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	kDefaultSession
Type:	`SessionControlType`
Syntax:	`constexpr SessionControlType kDefaultSession = 0x01;`
Description:	Default session according to IS0 14229-1.

⌋

8.8.2.2 kExtendedDiagnosticSession

[SWS_DM_01280] Definition of API variable ara::diag::kExtendedDiagnosticSession

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	kExtendedDiagnosticSession
Type:	`SessionControlType`
Syntax:	`constexpr SessionControlType kExtendedDiagnosticSession = 0x03;`
Description:	Extended diagnostic session according to IS0 14229-1.

⌋

8.8.2.3 kLocked

[SWS_DM_01282] Definition of API variable ara::diag::kLocked

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	kLocked
Type:	`SecurityLevelType`
Syntax:	`constexpr SecurityLevelType kLocked = 0x00;`
Description:	Security level locked.

⌋

8.8.2.4 kProgrammingSession

[SWS_DM_01279] Definition of API variable ara::diag::kProgrammingSession

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	kProgrammingSession
Type:	`SessionControlType`
Syntax:	`constexpr SessionControlType kProgrammingSession = 0x02;`
Description:	Programming session according to IS0 14229-1.

⌋

8.8.2.5 kSafetySystemDiagnosticSession

[SWS_DM_01281] Definition of API variable ara::diag::kSafetySystemDiagnostic Session

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/conversation.h"
Scope:	`namespace ara::diag`
Symbol:	kSafetySystemDiagnosticSession
Type:	`SessionControlType`
Syntax:	`constexpr SessionControlType kSafetySystemDiagnosticSession = 0x04;`
Description:	Safety system diagnostic session according to IS0 14229-1.

⌋

8.8.3 Class: Conversation

[SWS_DM_00693] Definition of API class ara::diag::Conversation

Upstream requirements: RS_Diag_04166, RS_Diag_04169

⌈

Kind:	class
Header file:	#include "ara/diag/conversation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Conversation
Syntax:	`class Conversation final {...};`
Description:	Conversation interface.

⌋

8.8.3.1 Public Member Functions

8.8.3.1.1 Member Functions

8.8.3.1.1.1 GetActivityStatus

[SWS_DM_00694] Definition of API function ara::diag::Conversation::GetActivityStatus

Upstream requirements: RS_AP_00139, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< ActivityStatusType > GetActivityStatus () noexcept;`	
Return value:	ara::core::Result< ActivityStatusType >	the activity status of the conversation
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Represents the status of an active conversation.	

⌋

8.8.3.1.1.2 GetAllConversations

[SWS_DM_00782] Definition of API function ara::diag::Conversation::GetAllConversations

Upstream requirements: RS_Diag_04166, RS_Diag_04169, RS_Diag_04209

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`static ara::core::Vector< std::reference_wrapper< Conversation > > GetAllConversations () noexcept;`	
Return value:	ara::core::Vector< std::reference_wrapper< Conversation > >	a vector of all possibe Conversation objects
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get all possible conversations.	

⌋

8.8.3.1.1.3 GetConversation

[SWS_DM_00692] Definition of API function ara::diag::Conversation::GetConversation

Upstream requirements: RS_AP_00139, RS_Diag_04169, RS_Diag_04170

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`static ara::core::Result< Conversation & > GetConversation (const MetaInfo &metaInfo) noexcept;`	
Parameters (in):	metaInfo	contains additional meta information
Return value:	ara::core::Result< Conversation & >	Conversation object or error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get one conversation based on given MetaInfo.	

⌋

8.8.3.1.1.4 GetConversationIdentifier

[SWS_DM_00700] Definition of API function ara::diag::Conversation::GetConversationIdentifier

Upstream requirements: RS_AP_00139, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< ConversationIdentifierType > GetConversation Identifier () noexcept;`	
Return value:	ara::core::Result< ConversationIdentifier Type >	the conversation information
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Getter for the current identification properties of the active conversation.	

⌋

8.8.3.1.1.5 GetCurrentActiveConversations

[SWS_DM_00783] Definition of API function ara::diag::Conversation::GetCurrentActiveConversations

Upstream requirements: RS_Diag_04166, RS_Diag_04169, RS_Diag_04209

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`static ara::core::Vector< std::reference_wrapper< Conversation > > GetCurrentActiveConversations () noexcept;`	
Return value:	ara::core::Vector< std::reference_wrapper< Conversation > >	a vector of all currently active (GetActivityStatus() == kActive) Conversation objects
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get all currently active conversations.	

⌋

8.8.3.1.1.6 GetDiagnosticSecurityLevel

[SWS_DM_00698] Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevel

Upstream requirements: RS_AP_00139, RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< SecurityLevelType > GetDiagnosticSecurityLevel () noexcept;`	
Return value:	ara::core::Result< SecurityLevelType >	the current SecurityLevel
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Represents the current active diagnostic SecurityLevel of an active conversation.	

⌋

8.8.3.1.1.7 GetDiagnosticSecurityLevelShortName

[SWS_DM_00708] Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevelShortName

Upstream requirements: RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< ara::core::StringView > GetDiagnosticSecurityLevelShortName (SecurityLevelType securityLevel) noexcept;`	
Parameters (in):	securityLevel	Security level enum the shortname shall be returned for.
Return value:	ara::core::Result< ara::core::StringView >	ara::core::Result<ara::core::StringView> the SecurityLevel as short Name; DiagnosticSecurityLevel.shortName
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		The requested security level is invalid
Description:	Converts the given diagnostic SecurityLevel into the ShortName.	

⌋

8.8.3.1.1.8 GetDiagnosticSession

[SWS_DM_00696] Definition of API function ara::diag::Conversation::GetDiagnosticSession

Upstream requirements: RS_AP_00139, RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< SessionControlType > GetDiagnosticSession () noexcept;`	
Return value:	ara::core::Result< SessionControlType >	the current session
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Represents the current active diagnostic session of an active conversation.	

⌋

8.8.3.1.1.9 GetDiagnosticSessionShortName

[SWS_DM_00707] Definition of API function ara::diag::Conversation::GetDiagnosticSessionShortName

Upstream requirements: RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< ara::core::StringView > GetDiagnosticSessionShortName (SessionControlType session) noexcept;`	
Parameters (in):	session	Diagnostic session the shortname shall be returned for.
Return value:	ara::core::Result< ara::core::StringView >	ara::core::Result<ara::core::StringView> the session as short Name; DiagnosticSession.shortName
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		The requested session is invalid

▽

△

Description:	Converts the given diagnostic session into the ShortName.

⌋

8.8.3.1.1.10 ResetToDefaultSession

[SWS_DM_00701] Definition of API function ara::diag::Conversation::ResetToDefaultSession

Upstream requirements: RS_Diag_04006, RS_Diag_04166, RS_Diag_04169, RS_Diag_04209

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< void > ResetToDefaultSession () noexcept;`	
Return value:	ara::core::Result< void >	void on success or error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Method to reset the current session to default session.	

⌋

8.8.3.1.1.11 SetActivityNotifier

[SWS_DM_00695] Definition of API function ara::diag::Conversation::SetActivityNotifier

Status: DRAFT

Upstream requirements: RS_AP_00139, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< void > SetActivityNotifier (std::function< void( ActivityStatusType)> notifier) noexcept;`	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	void when the registering went fine or error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Register a notifier function which is called if the activity is changed. A consecutive call of this method will overwrite the previous registered notifier.

⌋

8.8.3.1.1.12 SetDiagnosticSessionNotifier

[SWS_DM_00697] Definition of API function ara::diag::Conversation::SetDiagnosticSessionNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	`class ara::diag::Conversation`	
Syntax:	`ara::core::Result< void > SetDiagnosticSessionNotifier (std::function< void(SessionControlType)> notifier) noexcept;`	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	void when the registering went fine or error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Register a notifier function which is called if the Session is changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.8.3.1.1.13 SetSecurityLevelNotifier

[SWS_DM_00699] Definition of API function ara::diag::Conversation::SetSecurityLevelNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04169, RS_Diag_04208

⌈

Kind:	function
Header file:	#include "ara/diag/conversation.h"
Scope:	`class ara::diag::Conversation`
Syntax:	`ara::core::Result< void > SetSecurityLevelNotifier (std::function< void(SecurityLevelType)> notifier) noexcept;`

▽

△

Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	void when the registering went fine or error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Register a notifier function which is called if the SecurityLevel is changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.8.4 Struct: ConversationIdentifierType

[SWS_DM_00691] Definition of API class ara::diag::Conversation::ConversationIdentifierType

Upstream requirements: RS_Diag_04166, RS_Diag_04169, RS_Diag_04209

⌈

Kind:	struct
Header file:	#include "ara/diag/conversation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::Conversation`
Symbol:	ConversationIdentifierType
Syntax:	`struct ConversationIdentifierType {...};`
Description:	Properties allowing an identification of the conversation.

⌋

8.9 Header: ara/diag/data_transfer.h

8.9.1 Non-Member Types

8.9.1.1 Enumeration: DataTransferExitType

[SWS_DM_01538] Definition of API enum ara::diag::DataTransferExitType

Upstream requirements: RS_Diag_04135

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/data_transfer.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DataTransferExitType	
Underlying type:	ara::core::Byte	
Syntax:	`enum class DataTransferExitType : ara::core::Byte {...};`	
Values:	kkAcknowledge	The file transfer finished.
	kAbort	The file transfer has been aborted.
Description:	Determines transfer exit signal type.	

⌋

8.9.2 Class: DataTransferReadByPullHandler

[SWS_DM_01506] Definition of API class ara::diag::DataTransferReadByPullHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferReadByPullHandler
Syntax:	`class DataTransferReadByPullHandler {...};`
Description:	Handles data transfers initiated by RequestReadFile/-Directory with data pulled by AraDiag from the implementation.

⌋

8.9.2.1 Public Member Functions

8.9.2.1.1 Special Member Functions

8.9.2.1.1.1 Move Constructor

[SWS_DM_01508] Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPullHandler`	
Syntax:	`DataTransferReadByPullHandler (DataTransferReadByPullHandler &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.2.1.1.2 Copy Constructor

[SWS_DM_01509] Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPullHandler`
Syntax:	`DataTransferReadByPullHandler (DataTransferReadByPullHandler const &)=delete;`
Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .

⌋

8.9.2.1.1.3 Destructor

[SWS_DM_01510] Definition of API function ara::diag::DataTransferReadByPullHandler::~DataTransferReadByPullHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPullHandler`
Syntax:	`virtual ~DataTransferReadByPullHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before ExitRead() is called

⌋

8.9.2.1.2 Member Functions

8.9.2.1.2.1 ExitRead

[SWS_DM_01514] Definition of API function ara::diag::DataTransferReadByPullHandler::ExitRead

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPullHandler`	
Syntax:	`virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span< const ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Vector< ara::core::Byte > >=0;`	
Parameters (in):	exit_type	Specifies the exit reason
	transfer_request_parameter_record	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	meta_info	contains additional meta information

▽

△

	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_ data will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Exits an ongoing data transfer session.	

⌋

8.9.2.1.2.2 Read

[SWS_DM_01513] Definition of API function ara::diag::DataTransferReadByPull Handler::Read

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPullHandler`	
Syntax:	`virtual auto Read (ara::core::Span< const ara::core::Byte > response_ data, MetaInfo const &meta_info, CancellationHandler cancellation_ handler) noexcept -> ara::core::Future< std::uint32_t >=0;`	
Parameters (in):	response_data	The view over a pre-allocated by AraDiag memory to write in this data chunk, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< std::uint32_t >=0	A Future, which either gets readied to actual amount of data provided in the Span (for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in response_data will be placed after block SequenceCounter as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Reads data chunk from the application. The first or subsequent file/directory info content chunk to be sent back to the UDS client will be composed by the application and copied into the provided by AraDiag Span with up to the requested Span size.

⌋

8.9.2.1.2.3 operator=

[SWS_DM_01511] Definition of API function ara::diag::DataTransferReadByPull Handler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPullHandler`	
Syntax:	`auto operator= (DataTransferReadByPullHandler &&other) & noexcept -> DataTransferReadByPullHandler & =default;`	
Parameters (out):	other	The other object
Return value:	DataTransferReadByPull Handler & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.9.2.1.2.4 operator=

[SWS_DM_01512] Definition of API function ara::diag::DataTransferReadByPull Handler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPullHandler`
Syntax:	`auto operator= (DataTransferReadByPullHandler const &) -> DataTransfer ReadByPullHandler &=delete;`

▽

△

Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session.

⌋

8.9.2.2 Protected Member Functions

8.9.2.2.1 Special Member Functions

8.9.2.2.1.1 Default Constructor

[SWS_DM_01507] Definition of API function ara::diag::DataTransferReadByPull Handler::DataTransferReadByPullHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPullHandler`
Syntax:	`DataTransferReadByPullHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructs an instance of this class.
Visibility:	protected

⌋

8.9.3 Class: DataTransferReadByPushHandler

[SWS_DM_01515] Definition of API class ara::diag::DataTransferReadByPush Handler

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferReadByPushHandler

▽

△

Syntax:	`class DataTransferReadByPushHandler {...};`
Description:	Handles data transfers initiated by RequestReadFile/-Directory with data pushed by the implementation.

⌋

8.9.3.1 Public Member Functions

8.9.3.1.1 Special Member Functions

8.9.3.1.1.1 Move Constructor

[SWS_DM_01517] Definition of API function ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPushHandler`	
Syntax:	`DataTransferReadByPushHandler (DataTransferReadByPushHandler &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.3.1.1.2 Copy Constructor

[SWS_DM_01518] Definition of API function ara::diag::DataTransferReadByPush Handler::DataTransferReadByPushHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPushHandler`
Syntax:	`DataTransferReadByPushHandler (DataTransferReadByPushHandler const &)=delete;`
Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .

⌋

8.9.3.1.1.3 Destructor

[SWS_DM_01519] Definition of API function ara::diag::DataTransferReadByPush Handler::~DataTransferReadByPushHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPushHandler`
Syntax:	`virtual ~DataTransferReadByPushHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before ExitRead() is called

⌋

8.9.3.1.2 Member Functions

8.9.3.1.2.1 ExitRead

[SWS_DM_01523] Definition of API function ara::diag::DataTransferReadByPush Handler::ExitRead

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPushHandler`	
Syntax:	`virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span< const ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Vector< ara::core::Byte > >=0;`	
Parameters (in):	exit_type	Specifies the exit reason
	transfer_request_parameter_record	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	meta_info	contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_data will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Exits an ongoing data transfer session.	

⌋

8.9.3.1.2.2 Read

[SWS_DM_01522] Definition of API function ara::diag::DataTransferReadByPush Handler::Read

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPushHandler`	
Syntax:	`virtual auto Read (std::size_t recommended_number_bytes_to_return, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Vector< ara::core::Byte > >=0;`	
Parameters (in):	recommended_number_bytes_to_return	The recommended number of bytes to send back to DM in order to get optimum data throughput
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	A Future, which either gets readied to ReadPushOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in ReadPushOutput.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Reads data chunk from the application. The first or subsequent file/directory info content chunk to be sent back to the UDS client will be composed by the application and its ownership is passed as a result to AraDiag.	

⌋

8.9.3.1.2.3 operator=

[SWS_DM_01520] Definition of API function ara::diag::DataTransferReadByPush Handler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadByPushHandler`	
Syntax:	`auto operator= (DataTransferReadByPushHandler &&other) & noexcept -> DataTransferReadByPushHandler & =default;`	
Parameters (out):	other	The other object
Return value:	DataTransferReadBy PushHandler & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.9.3.1.2.4 operator=

[SWS_DM_01521] Definition of API function ara::diag::DataTransferReadByPush Handler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPushHandler`
Syntax:	`auto operator= (DataTransferReadByPushHandler const &) -> DataTransfer ReadByPushHandler &=delete;`
Description:	Handlers shall not be assignable since only one way usage i.e. per single file transfer session.

⌋

8.9.3.2 Protected Member Functions

8.9.3.2.1 Special Member Functions

8.9.3.2.1.1 Default Constructor

[SWS_DM_01516] Definition of API function ara::diag::DataTransferReadByPush Handler::DataTransferReadByPushHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadByPushHandler`
Syntax:	`DataTransferReadByPushHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructs an instance of this class.
Visibility:	protected

⌋

8.9.4 Class: DataTransferReadSession

[SWS_DM_01548] Definition of API class ara::diag::DataTransferReadSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferReadSession
Syntax:	`class DataTransferReadSession final {...};`
Description:	Encapsulates all data transfer reading variants .

⌋

8.9.4.1 Public Member Functions

8.9.4.1.1 Special Member Functions

8.9.4.1.1.1 Move Constructor

[SWS_DM_01551] Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSession`	
Syntax:	`DataTransferReadSession (DataTransferReadSession &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.4.1.1.2 Default Constructor

[SWS_DM_01549] Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSession`
Syntax:	`DataTransferReadSession ()=delete;`
Description:	No default construction allowed.

⌋

8.9.4.1.1.3 Copy Constructor

[SWS_DM_01553] Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSession`
Syntax:	`DataTransferReadSession (DataTransferReadSession const &)=delete;`
Description:	DataTransferReadSession shall be a single not copy-able instance. .

⌋

8.9.4.1.1.4 Destructor

[SWS_DM_01550] Definition of API function ara::diag::DataTransferReadSession::~DataTransferReadSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSession`
Syntax:	`~DataTransferReadSession () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class.

⌋

8.9.4.1.2 Member Functions

8.9.4.1.2.1 operator=

[SWS_DM_01554] Definition of API function ara::diag::DataTransferReadSession::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSession`
Syntax:	`auto operator= (DataTransferReadSession const &) -> DataTransferReadSession &=delete;`
Description:	DataTransferReadSession shall be a single not assignable instance.

⌋

8.9.4.1.2.2 operator=

[SWS_DM_01552] Definition of API function ara::diag::DataTransferReadSession::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSession`	
Syntax:	`auto operator= (DataTransferReadSession &&other) & noexcept -> DataTransferReadSession & =default;`	
Parameters (out):	other	The other object
Return value:	DataTransferReadSession & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.9.5 Class: DataTransferReadSharedDataHandler

[SWS_DM_01497] Definition of API class ara::diag::DataTransferReadSharedDataHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferReadSharedDataHandler
Syntax:	`class DataTransferReadSharedDataHandler {...};`
Description:	Handles data transfers initiated by RequestReadFile/-Directory with shared data from the implementation.

⌋

8.9.5.1 Public Member Functions

8.9.5.1.1 Special Member Functions

8.9.5.1.1.1 Move Constructor

[SWS_DM_01500] Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`	
Syntax:	`DataTransferReadSharedDataHandler (DataTransferReadSharedDataHandler &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.5.1.1.2 Default Constructor

[SWS_DM_01498] Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`
Syntax:	`DataTransferReadSharedDataHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructs an instance of this class.

⌋

8.9.5.1.1.3 Copy Constructor

[SWS_DM_01501] Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`
Syntax:	`DataTransferReadSharedDataHandler (DataTransferReadSharedDataHandler const &)=delete;`
Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .

⌋

8.9.5.1.1.4 Destructor

[SWS_DM_01499] Definition of API function ara::diag::DataTransferReadSharedDataHandler::~DataTransferReadSharedDataHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`
Syntax:	`virtual ~DataTransferReadSharedDataHandler () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before ExitRead() is called

⌋

8.9.5.1.2 Member Functions

8.9.5.1.2.1 ExitRead

[SWS_DM_01505] Definition of API function ara::diag::DataTransferReadSharedDataHandler::ExitRead

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`	
Syntax:	`virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span< const ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Vector< ara::core::Byte > >=0;`	
Parameters (in):	exit_type	Specifies the exit reason
	transfer_request_parameter_record	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	meta_info	contains additional meta information

▽

△

	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_ data will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Exits an ongoing data transfer session.	

⌋

8.9.5.1.2.2 Read

[SWS_DM_01504] Definition of API function ara::diag::DataTransferReadSharedDataHandler::Read

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`	
Syntax:	`virtual auto Read (ReleaseHandler release_handler, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Span< ara::core::Byte > >=0;`	
Parameters (in):	release_handler	The release shared resource handler to be stored and used by the application until a new one is passed for the same meta_info (UDS client)
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Span< ara::core::Byte > >=0	A Future, which either gets readied to ReadSharedOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in ReleaseSharedHandler.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Provides thw whole data to be read out from the application in form of a shared data view. The whole file/directory info content to be sent back to the UDS client will be allocated by the application and its shared ownership (view over the memory location) is passed as a result to AraDiag. The allocated memory must be kept until the ReleaseHandler's notifier is invoked or the status polled, signalling that the memory addressed by the Span now can safely be released.

⌋

8.9.5.1.2.3 operator=

[SWS_DM_01502] Definition of API function ara::diag::DataTransferReadSharedDataHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`	
Syntax:	`auto operator= (DataTransferReadSharedDataHandler &&other) & noexcept -> DataTransferReadSharedDataHandler & =default;`	
Parameters (out):	other	The other object
Return value:	DataTransferRead SharedDataHandler & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.9.5.1.2.4 operator=

[SWS_DM_01503] Definition of API function ara::diag::DataTransferReadSharedDataHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferReadSharedDataHandler`

▽

△

Syntax:	`auto operator= (DataTransferReadSharedDataHandler const &) -> DataTransferReadSharedDataHandler &=delete;`
Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session.

⌋

8.9.6 Class: DataTransferWriteHandler

[SWS_DM_01539] Definition of API class ara::diag::DataTransferWriteHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferWriteHandler
Syntax:	`class DataTransferWriteHandler {...};`
Description:	Handles data transfers initiated by AddFile, ResumeFile or ReplaceFile ModeOfOperation handling per requested RequestWriteFile.

⌋

8.9.6.1 Public Member Functions

8.9.6.1.1 Special Member Functions

8.9.6.1.1.1 Copy Constructor

[SWS_DM_01544] Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteHandler`
Syntax:	`DataTransferWriteHandler (DataTransferWriteHandler const &)=delete;`

▽

△

Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .

⌋

8.9.6.1.1.2 Move Constructor

[SWS_DM_01542] Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferWriteHandler`	
Syntax:	`DataTransferWriteHandler (DataTransferWriteHandler &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.6.1.1.3 Destructor

[SWS_DM_01541] Definition of API function ara::diag::DataTransferWriteHandler::~DataTransferWriteHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteHandler`
Syntax:	`virtual ~DataTransferWriteHandler () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before ExitWrite() is called

⌋

8.9.6.1.2 Member Functions

8.9.6.1.2.1 ExitWrite

[SWS_DM_01547] Definition of API function ara::diag::DataTransferWriteHandler::ExitWrite

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferWriteHandler`	
Syntax:	`virtual auto ExitWrite (DataTransferExitType exitType, ara::core::Span< ara::core::Byte > transferRequestParameterRecord, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< ara::core::Vector< ara::core::Byte > >=0;`	
Parameters (in):	exitType	Specifies the exit reason
	transferRequest ParameterRecord	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_ data will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Exits an ongoing data transfer session.	

⌋

8.9.6.1.2.2 Write

[SWS_DM_01546] Definition of API function ara::diag::DataTransferWriteHandler::Write

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferWriteHandler`	
Syntax:	`virtual auto Write (ara::core::Span< ara::core::Byte > requestData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< void >=0;`	
Parameters (in):	requestData	The first or subsequent file content chunk received from the UDS client to be written into the file, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >=0	A Future, which either is of type void or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message).
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Appends a data chunk into application memory.	

⌋

8.9.6.1.2.3 operator=

[SWS_DM_01543] Definition of API function ara::diag::DataTransferWriteHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferWriteHandler`	
Syntax:	`auto operator= (DataTransferWriteHandler &&other) & noexcept -> DataTransferWriteHandler & =default;`	
Parameters (out):	other	The other object

▽

△

Return value:	DataTransferWrite Handler & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.9.6.1.2.4 operator=

[SWS_DM_01545] Definition of API function ara::diag::DataTransferWriteHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteHandler`
Syntax:	`auto operator= (DataTransferWriteHandler const &) -> DataTransferWriteHandler &=delete;`
Description:	Handlers shall not be copiable since only one way usage i.e. per single file transfer session.

⌋

8.9.6.2 Protected Member Functions

8.9.6.2.1 Special Member Functions

8.9.6.2.1.1 Default Constructor

[SWS_DM_01540] Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteHandler`

▽

△

Syntax:	`DataTransferWriteHandler ()=default;`
Thread Safety:	thread-safe
Description:	Constructs an instance of this class.
Visibility:	protected

⌋

8.9.7 Class: DataTransferWriteSession

[SWS_DM_01555] Definition of API class ara::diag::DataTransferWriteSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/data_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DataTransferWriteSession
Syntax:	`class DataTransferWriteSession final {...};`
Description:	Encapsulates all data transfer writing variants .

⌋

8.9.7.1 Public Member Functions

8.9.7.1.1 Special Member Functions

8.9.7.1.1.1 Move Constructor

[SWS_DM_01557] Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteSession`

▽

△

Syntax:	`DataTransferWriteSession (DataTransferWriteSession &&other) noexcept=default;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of this class.	

⌋

8.9.7.1.1.2 Default Constructor

[SWS_DM_01339] Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteSession`
Syntax:	`DataTransferWriteSession ()=delete;`
Description:	No default construction allowed.

⌋

8.9.7.1.1.3 Copy Constructor

[SWS_DM_01559] Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteSession`
Syntax:	`DataTransferWriteSession (DataTransferWriteSession const &)=delete;`
Description:	DataTransferWriteSession shall be a single not copy-able instance. .

⌋

8.9.7.1.1.4 Destructor

[SWS_DM_01556] Definition of API function ara::diag::DataTransferWriteSession::~DataTransferWriteSession

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteSession`
Syntax:	`~DataTransferWriteSession () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructs an instance of this class.

⌋

8.9.7.1.2 Member Functions

8.9.7.1.2.1 operator=

[SWS_DM_01560] Definition of API function ara::diag::DataTransferWriteSession::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	`class ara::diag::DataTransferWriteSession`
Syntax:	`auto operator= (DataTransferWriteSession const &) -> DataTransferWriteSession &=delete;`
Description:	DataTransferWriteSession shall be a single not assignable instance.

⌋

8.9.7.1.2.2 operator=

[SWS_DM_01558] Definition of API function ara::diag::DataTransferWriteSession::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	`class ara::diag::DataTransferWriteSession`	
Syntax:	`auto operator= (DataTransferWriteSession &&other) & noexcept -> DataTransferWriteSession & =default;`	
Parameters (out):	other	The other object
Return value:	DataTransferWrite Session & =default	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of this class.	

⌋

8.10 Header: ara/diag/diag_error_domain.h

8.10.1 Non-Member Types

8.10.1.1 Enumeration: DiagErrc

[SWS_DM_00514] Definition of API enum ara::diag::DiagErrc

Upstream requirements: RS_AP_00119, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/diag_error_domain.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DiagErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class DiagErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kAlreadyOffered= 101	The service is already offered.
	kConfiguration Mismatch= 102	monitor configuration does not match dext

▽

△

	kDebouncing Configuration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
	kReportIgnored= 104	Enable Conditions disabled, OC not started, ...
	kInvalidArgument= 105	e.g. kPreFailed with internal debouncing
	kNotOffered= 106	Offer not called before reporting.
	kNoSuchDTC= 108	No DTC available.
	kBusy= 109	Interface is busy with processing.
	kFailed= 110	Failed to process.
	kMemoryError= 111	A memory error occurred during processing.
	kWrongDtc= 112	A wrong DTC number was requested.
	kRejected= 113	Requested operation was rejected due to StateManagements/ machines internal state.
	kResetTypeNot Supported= 114	The requested Diagnostic reset type is not supported by the Diagnostic Address instance.
	kRequestFailed= 115	Diagnostic request could not be performed successfully.
	kCustomResetTypeNot Supported= 116	The requested Diagnostic custom reset type is not supported by the Diagnostic Address instance.
	kSuppressionIgnored= 117	In case a suppression of a DTC is requested, but the conditions are not met.
	kServiceNotAvailable= 118	The call cannot be executed, because essential DM functionality is currently not available.
Description:	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.	

⌋

8.10.1.2 Enumeration: DiagOfferErrc

[SWS_DM_00559] Definition of API enum ara::diag::DiagOfferErrc

Upstream requirements: RS_AP_00119, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/diag_error_domain.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DiagOfferErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class DiagOfferErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kAlreadyOffered= 101	The service is already offered.
	kConfiguration Mismatch= 102	monitor configuration does not match dext

▽

△

	kDebouncing Configuration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
Description:	The DiagOfferErrc enumeration defines the error codes for the DiagOfferErrorDomain.	

⌋

8.10.1.3 Enumeration: DiagReportingErrc

[SWS_DM_00560] Definition of API enum ara::diag::DiagReportingErrc

Upstream requirements: RS_AP_00119, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/diag_error_domain.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DiagReportingErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class DiagReportingErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kAlreadyOffered= 101	The service is already offered.
	kConfiguration Mismatch= 102	monitor configuration does not match dext
	kDebouncing Configuration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
	kReportIgnored= 104	Enable Conditions disabled, OC not started, ...
	kInvalidArgument= 105	e.g. kPreFailed with internal debouncing
	kNotOffered= 106	Offer not called before reporting.
	kNoSuchDTC= 108	No mapped DTC exists for the requested event.
Description:	The DiagReportingErrc enumeration defines the error codes for the DiagReportingErrorDomain. .	

⌋

8.10.2 Non-Member Functions

8.10.2.1 Other

8.10.2.1.1 GetDiagDomain

[SWS_DM_00524] Definition of API function ara::diag::GetDiagDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetDiagDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	reference to the DiagErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagErrorDomain instance.	

⌋

8.10.2.1.2 GetDiagOfferDomain

[SWS_DM_01003] Definition of API function ara::diag::GetDiagOfferDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetDiagOfferDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	reference to the DiagOfferErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagErrorDomain instance.	

⌋

8.10.2.1.3 GetDiagReportingDomain

[SWS_DM_01004] Definition of API function ara::diag::GetDiagReportingDomain

Upstream requirements: RS_AP_00119, RS_AP_00125, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetDiagReportingDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	reference to the DiagOfferErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagReportingErrorDomain instance.	

⌋

8.10.2.1.4 MakeErrorCode

[SWS_DM_01006] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (DiagReportingErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Create a new ErrorCode for DiagReportingErrorDomain with the given support data type.	

⌋

8.10.2.1.5 MakeErrorCode

[SWS_DM_00525] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (DiagErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Create a new ErrorCode for DiagErrorDomain with the given support data type.	

⌋

8.10.2.1.6 MakeErrorCode

[SWS_DM_01005] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (DiagOfferErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Create a new ErrorCode for DiagOfferErrorDomain with the given support data type.	

⌋

8.10.3 Class: DiagErrorDomain

[SWS_DM_00517] Definition of API class ara::diag::DiagErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class DiagErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::diag::DiagErrorDomain` in [SWS_CORE_90023]
Description:	Error domain for diagnostic errors.

⌋

8.10.3.1 Public Member Types

8.10.3.1.1 Type Alias: Errc

[SWS_DM_00518] Definition of API type ara::diag::DiagErrorDomain::Errc

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = DiagErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.10.3.1.2 Type Alias: Exception

[SWS_DM_00519] Definition of API type ara::diag::DiagErrorDomain::Exception

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = DiagException;`
Description:	Alias for the exception base class.

⌋

8.10.3.2 Public Member Functions

8.10.3.2.1 Special Member Functions

8.10.3.2.1.1 Default Constructor

[SWS_DM_00520] Definition of API function ara::diag::DiagErrorDomain::DiagErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagErrorDomain`
Syntax:	`constexpr DiagErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

⌋

8.10.3.2.2 Member Functions

8.10.3.2.2.1 Message

[SWS_DM_00522] Definition of API function ara::diag::DiagErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.10.3.2.2.2 Name

[SWS_DM_00521] Definition of API function ara::diag::DiagErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"Diag"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

⌋

8.10.3.2.2.3 ThrowAsException

[SWS_DM_00523] Definition of API function ara::diag::DiagErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given ErrorCode. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.10.4 Class: DiagException

[SWS_DM_00515] Definition of API class ara::diag::DiagException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagException
Base class:	ara::core::Exception
Syntax:	`class DiagException : public ara::core::Exception {...};`
Description:	Exception type thrown by Diag classes.

⌋

8.10.4.1 Public Member Functions

8.10.4.1.1 Constructors

8.10.4.1.1.1 DiagException

[SWS_DM_00516] Definition of API function ara::diag::DiagException::DiagException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagException`	
Syntax:	`explicit DiagException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagException from an ErrorCode.	

⌋

8.10.5 Class: DiagOfferErrorDomain

[SWS_DM_00989] Definition of API class ara::diag::DiagOfferErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagOfferErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class DiagOfferErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::diag::DiagOfferErrorDomain` in [SWS_CORE_90023]
Description:	Error domain for diagnostic offer errors.

⌋

8.10.5.1 Public Member Types

8.10.5.1.1 Type Alias: Errc

[SWS_DM_00990] Definition of API type ara::diag::DiagOfferErrorDomain::Errc

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagOfferErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = DiagOfferErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.10.5.1.2 Type Alias: Exception

[SWS_DM_00991] Definition of API type ara::diag::DiagOfferErrorDomain::Exception

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagOfferErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = DiagException;`
Description:	Alias for the exception base class.

⌋

8.10.5.2 Public Member Functions

8.10.5.2.1 Special Member Functions

8.10.5.2.1.1 Default Constructor

[SWS_DM_00992] Definition of API function ara::diag::DiagOfferErrorDomain::DiagOfferErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagOfferErrorDomain`
Syntax:	`constexpr DiagOfferErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

⌋

8.10.5.2.2 Member Functions

8.10.5.2.2.1 Message

[SWS_DM_00994] Definition of API function ara::diag::DiagOfferErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagOfferErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.10.5.2.2.2 Name

[SWS_DM_00993] Definition of API function ara::diag::DiagOfferErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagOfferErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"DiagOffer"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

⌋

8.10.5.2.2.3 ThrowAsException

[SWS_DM_00995] Definition of API function ara::diag::DiagOfferErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagOfferErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given ErrorCode. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.10.6 Class: DiagOfferException

[SWS_DM_00985] Definition of API class ara::diag::DiagOfferException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagOfferException
Base class:	ara::core::Exception
Syntax:	`class DiagOfferException : public ara::core::Exception {...};`
Description:	Exception type thrown by Diag classes.

⌋

8.10.6.1 Public Member Functions

8.10.6.1.1 Constructors

8.10.6.1.1.1 DiagOfferException

[SWS_DM_00986] Definition of API function ara::diag::DiagOfferException::DiagOfferException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagOfferException`	
Syntax:	`explicit DiagOfferException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagException from an ErrorCode.	

⌋

8.10.7 Class: DiagReportingErrorDomain

[SWS_DM_00996] Definition of API class ara::diag::DiagReportingErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagReportingErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class DiagReportingErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::diag::DiagReportingErrorDomain` in [SWS_CORE_90023]
Description:	Error domain for diagnostic reporting errors.

⌋

8.10.7.1 Public Member Types

8.10.7.1.1 Type Alias: Errc

[SWS_DM_00997] Definition of API type ara::diag::DiagReportingErrorDomain::Errc

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagReportingErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = DiagReportingErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.10.7.1.2 Type Alias: Exception

[SWS_DM_00998] Definition of API type ara::diag::DiagReportingErrorDomain::Exception

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagReportingErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = DiagReportingException;`
Description:	Alias for the exception base class.

⌋

8.10.7.2 Public Member Functions

8.10.7.2.1 Special Member Functions

8.10.7.2.1.1 Default Constructor

[SWS_DM_00999] Definition of API function ara::diag::DiagReportingErrorDomain::DiagReportingErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	`class ara::diag::DiagReportingErrorDomain`
Syntax:	`constexpr DiagReportingErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

⌋

8.10.7.2.2 Member Functions

8.10.7.2.2.1 Message

[SWS_DM_01001] Definition of API function ara::diag::DiagReportingErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagReportingErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.10.7.2.2.2 Name

[SWS_DM_01000] Definition of API function ara::diag::DiagReportingErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagReportingErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"DiagReporting"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

⌋

8.10.7.2.2.3 ThrowAsException

[SWS_DM_01002] Definition of API function ara::diag::DiagReportingErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagReportingErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given ErrorCode. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.10.8 Class: DiagReportingException

[SWS_DM_00987] Definition of API class ara::diag::DiagReportingException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagReportingException
Base class:	ara::core::Exception
Syntax:	`class DiagReportingException : public ara::core::Exception {...};`
Description:	Exception type thrown by Diag classes.

⌋

8.10.8.1 Public Member Functions

8.10.8.1.1 Constructors

8.10.8.1.1.1 DiagReportingException

[SWS_DM_00988] Definition of API function ara::diag::DiagReportingException::DiagReportingException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	`class ara::diag::DiagReportingException`	
Syntax:	`explicit DiagReportingException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagException from an ErrorCode.	

⌋

8.11 Header: ara/diag/diag_uds_nrc_error_domain.h

8.11.1 Non-Member Types

8.11.1.1 Enumeration: DiagUdsNrcErrc

[SWS_DM_00526] Definition of API enum ara::diag::DiagUdsNrcErrc

Upstream requirements: RS_AP_00119, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DiagUdsNrcErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class DiagUdsNrcErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kGeneralReject= 0x10	According to ISO.

▽

△

	kServiceNotSupported= 0x11	According to ISO.
	kSubfunctionNot Supported= 0x12	According to ISO.
	kIncorrectMessage LengthOrInvalidFormat= 0x13	According to ISO.
	kResponseTooLong= 0x14	According to ISO.
	kBusyRepeatRequest= 0x21	According to ISO.
	kConditionsNotCorrect= 0x22	According to ISO.
	kRequestSequence Error= 0x24	According to ISO.
	kNoResponseFrom SubnetComponent= 0x25	According to ISO.
	kFailurePrevents ExecutionOfRequested Action= 0x26	According to ISO.
	kRequestOutOfRange= 0x31	According to ISO.
	kSecurityAccessDenied= 0x33	According to ISO.
	kAuthentication Required= 0x34	According to ISO.
	kInvalidKey= 0x35	According to ISO.
	kExceedNumberOf Attempts= 0x36	According to ISO.
	kRequiredTimeDelayNot Expired= 0x37	According to ISO.
	kCertificateVerification FailedInvalidTime Period= 0x50	According to ISO.
	kCertificateVerification FailedInvalidSignature= 0x51	According to ISO.
	kCertificateVerification FailedInvalidChainOf Trust= 0x52	According to ISO.
	kCertificateVerification FailedInvalidType= 0x53	According to ISO.
	kCertificateVerification FailedInvalidFormat= 0x54	According to ISO.
	kCertificateVerification FailedInvalidContent= 0x55	According to ISO.
	kCertificateVerification FailedInvalidScope= 0x56	According to ISO.
	kCertificateVerification FailedInvalidCertificate Revoked= 0x57	According to ISO.
	kOwnershipVerification Failed= 0x58	According to ISO.

▽

△

	kChallengeCalculation Failed= 0x59	According to ISO.
	kSettingAccessRights Failed= 0x5A	According to ISO.
	kSessionKeyCreation DerivationFailed= 0x5B	According to ISO.
	kConfigurationData UsageFailed= 0x5C	According to ISO.
	kDeAuthentication Failed= 0x5D	According to ISO.
	kUploadDownloadNot Accepted= 0x70	According to ISO.
	kTransferData Suspended= 0x71	According to ISO.
	kGeneralProgramming Failure= 0x72	According to ISO.
	kWrongBlockSequence Counter= 0x73	According to ISO.
	kSubFunctionNot SupportedInActive Session= 0x7E	According to ISO.
	kServiceNotSupportedIn ActiveSession= 0x7F	According to ISO.
	kRpmTooHigh= 0x81	According to ISO.
	kRpmTooLow= 0x82	According to ISO.
	kEngineIsRunning= 0x83	According to ISO.
	kEngineIsNotRunning= 0x84	According to ISO.
	kEngineRunTimeToo Low= 0x85	According to ISO.
	kTemperatureTooHigh= 0x86	According to ISO.
	kTemperatureTooLow= 0x87	According to ISO.
	kVehicleSpeedTooHigh= 0x88	According to ISO.
	kVehicleSpeedTooLow= 0x89	According to ISO.
	kThrottlePedalTooHigh= 0x8A	According to ISO.
	kThrottlePedalTooLow= 0x8B	According to ISO.
	kTransmissionRangeNot InNeutral= 0x8C	According to ISO.
	kTransmissionRangeNot InGear= 0x8D	According to ISO.
	kBrakeSwitchNot Closed= 0x8F	According to ISO.
	kShifterLeverNotInPark= 0x90	According to ISO.
	kTorqueConverterClutch Locked= 0x91	According to ISO.
	kVoltageTooHigh= 0x92	According to ISO.
	kVoltageTooLow= 0x93	According to ISO.

▽

△

	kResourceTemporarily NotAvailable= 0x94	According to ISO 14229-1 Table A.1.14229-1 Table A.1.
	kNoProcessingNo Response= 0xFF	Deviating from ISO - no further service processing and no response (silently ignore request message).
	kISO_RESERVED_ NRC= 0x01	Placeholder as starting number for valid NRCs.
Description:	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.	

⌋

8.11.2 Non-Member Functions

8.11.2.1 Other

8.11.2.1.1 GetDiagUdsNrcDomain

[SWS_DM_00536] Definition of API function ara::diag::GetDiagUdsNrcDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetDiagUdsNrcDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	reference to the DiagUdsNrcErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagUdsNrcErrorDomain instance.	

⌋

8.11.2.1.2 MakeErrorCode

[SWS_DM_00537] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (DiagUdsNrcErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an enumeration value from diag_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Create a new ErrorCode for DiagUdsNrcErrorDomain with the given support data type and message.	

⌋

8.11.3 Class: DiagUdsNrcErrorDomain

[SWS_DM_00529] Definition of API class ara::diag::DiagUdsNrcErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagUdsNrcErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class DiagUdsNrcErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::diag::DiagUdsNrcErrorDomain` in [SWS_CORE_90023]
Description:	Error domain for errors originating from several diagnostic classes.

⌋

8.11.3.1 Public Member Types

8.11.3.1.1 Type Alias: Errc

[SWS_DM_00530] Definition of API type ara::diag::DiagUdsNrcErrorDomain::Errc

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = DiagUdsNrcErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.11.3.1.2 Type Alias: Exception

[SWS_DM_00531] Definition of API type ara::diag::DiagUdsNrcErrorDomain::Exception

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = DiagUdsNrcException;`
Description:	Alias for the exception base class.

⌋

8.11.3.2 Public Member Functions

8.11.3.2.1 Special Member Functions

8.11.3.2.1.1 Default Constructor

[SWS_DM_00532] Definition of API function ara::diag::DiagUdsNrcErrorDomain::DiagUdsNrcErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`
Syntax:	`constexpr DiagUdsNrcErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

⌋

8.11.3.2.2 Member Functions

8.11.3.2.2.1 Message

[SWS_DM_00534] Definition of API function ara::diag::DiagUdsNrcErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.11.3.2.2.2 Name

[SWS_DM_00533] Definition of API function ara::diag::DiagUdsNrcErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"DiagUdsNrc"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

⌋

8.11.3.2.2.3 ThrowAsException

[SWS_DM_00535] Definition of API function ara::diag::DiagUdsNrcErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`class ara::diag::DiagUdsNrcErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given ErrorCode. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.11.4 Class: DiagUdsNrcException

[SWS_DM_00527] Definition of API class ara::diag::DiagUdsNrcException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagUdsNrcException
Base class:	ara::core::Exception
Syntax:	`class DiagUdsNrcException : public ara::core::Exception {...};`
Description:	Exception type thrown by Diag classes.

⌋

8.11.4.1 Public Member Functions

8.11.4.1.1 Constructors

8.11.4.1.1.1 DiagUdsNrcException

[SWS_DM_00528] Definition of API function ara::diag::DiagUdsNrcException::DiagUdsNrcException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	`class ara::diag::DiagUdsNrcException`	
Syntax:	`explicit DiagUdsNrcException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagException from an ErrorCode.	

⌋

8.12 Header: ara/diag/diagnostic_service_dynamic_access_list.h

8.12.1 Class: DiagnosticServiceDynamicAccessList

[SWS_DM_01156] Definition of API class ara::diag::DiagnosticServiceDynamicAccessList

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagnosticServiceDynamicAccessList
Syntax:	`class DiagnosticServiceDynamicAccessList final {...};`
Description:	Definition of the DiagnosticServiceDynamicAccessList class, which is used by the application to build a DynamicAccessList.

⌋

8.12.1.1 Public Member Functions

8.12.1.1.1 Special Member Functions

8.12.1.1.1.1 Copy Constructor

[SWS_DM_01159] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList const &other) noexcept;`	
Parameters (in):	other	Object to copy-construct from
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor of DiagnosticServiceDynamicAccessList.	

⌋

8.12.1.1.1.2 Default Constructor

[SWS_DM_01157] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`
Syntax:	`DiagnosticServiceDynamicAccessList () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructor of DiagnosticServiceDynamicAccessList .

⌋

8.12.1.1.1.3 Move Constructor

[SWS_DM_01160] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList &&other) noexcept;`	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of DiagnosticServiceDynamicAccessList.	

⌋

8.12.1.1.1.4 Destructor

[SWS_DM_01158] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::~DiagnosticServiceDynamicAccessList

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`
Syntax:	`~DiagnosticServiceDynamicAccessList () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticServiceDynamicAccessList .

⌋

8.12.1.1.2 Member Functions

8.12.1.1.2.1 MakeServiceBuilder

[SWS_DM_01165] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte String serviceHead) noexcept -> DynamicAccessListDiagServiceBuilder;`	
Parameters (in):	serviceHead	A string of bytes to start the DynamicAccess pattern-match
Return value:	DynamicAccessListDiag ServiceBuilder	An instance of a diagnostic service pattern builder
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the Application to construct a pattern for the DynamicAccessList using a string of bytes.	

⌋

8.12.1.1.2.2 MakeServiceBuilder

[SWS_DM_01164] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte sid) noexcept -> DynamicAccessListDiagServiceBuilder;`	
Parameters (in):	sid	The diagnostic service identifier
Return value:	DynamicAccessListDiagServiceBuilder	An instance of a diagnostic service pattern builder
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the Application to construct a pattern for the DynamicAccessList using only the SID.	

⌋

8.12.1.1.2.3 Reserve

[SWS_DM_01163] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::Reserve

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`void Reserve (std::size_t numberOfServiceHeads, std::size_t maxServiceHeadSize) noexcept;`	
Parameters (in):	numberOfServiceHeads	The number of diagnostic service patterns
	maxServiceHeadSize	The expected maximum number of diagnostic service bytes in a single pattern
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Preallocates memory for all service heads to fit into the DynamicAccessList. The preallocation can be just estimated and may calculate just the worst case of memory needed, not exact memory size needed for data numberOfServiceHeads.	

⌋

8.12.1.1.2.4 operator=

[SWS_DM_01162] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`auto operator= (DiagnosticServiceDynamicAccessList &&other) & noexcept -> DiagnosticServiceDynamicAccessList &;`	
Parameters (in):	other	Object to move-assign from.
Return value:	DiagnosticService DynamicAccessList &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of DiagnosticServiceDynamicAccessList.	

⌋

8.12.1.1.2.5 operator=

[SWS_DM_01161] Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Scope:	`class ara::diag::DiagnosticServiceDynamicAccessList`	
Syntax:	`auto operator= (DiagnosticServiceDynamicAccessList const &other) & noexcept -> DiagnosticServiceDynamicAccessList &;`	
Parameters (in):	other	Object to copy-assign from.
Return value:	DiagnosticService DynamicAccessList &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator of DiagnosticServiceDynamicAccessList.	

⌋

8.13 Header: ara/diag/download.h

8.13.1 Class: DownloadService

[SWS_DM_00784] Definition of API class ara::diag::DownloadService

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	class
Header file:	#include "ara/diag/download.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DownloadService
Syntax:	`class DownloadService {...};`
Description:	Download service interface.

⌋

8.13.1.1 Public Member Functions

8.13.1.1.1 Special Member Functions

8.13.1.1.1.1 Move Constructor

[SWS_DM_01650] Definition of API function ara::diag::DownloadService::DownloadService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`DownloadService (DownloadService &&) noexcept=delete;`
Description:	Move constructor of DownloadService.

⌋

8.13.1.1.1.2 Move Assignment Operator

[SWS_DM_01648] Definition of API function ara::diag::DownloadService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`DownloadService & operator= (DownloadService &&)=delete;`
Description:	Move assignment operator of DownloadService.

⌋

8.13.1.1.1.3 Destructor

[SWS_DM_00788] Definition of API function ara::diag::DownloadService::~DownloadService

Upstream requirements: RS_AP_00134, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`virtual ~DownloadService () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class DownloadService .

⌋

8.13.1.1.2 Constructors

8.13.1.1.2.1 DownloadService

[SWS_DM_00787] Definition of API function ara::diag::DownloadService::DownloadService

Upstream requirements: RS_AP_00137, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	`class ara::diag::DownloadService`	
Syntax:	`explicit DownloadService (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticDownloadInterface` needs to be typed by a `DiagnosticServiceGenericMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Class for an DownloadService.	

⌋

8.13.1.1.2.2 DownloadService

[SWS_DM_01649] Definition of API function ara::diag::DownloadService::DownloadService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`DownloadService (DownloadService &)=delete;`
Description:	DownloadService shall be a single not copy-able instance.

⌋

8.13.1.1.3 Member Functions

8.13.1.1.3.1 DownloadData

[SWS_DM_00790] Definition of API function ara::diag::DownloadService::DownloadData

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	`class ara::diag::DownloadService`	
Syntax:	`virtual ara::core::Future< OperationOutput > DownloadData (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	transferRequest ParameterRecord	data to be transferred (copied/downloaded to the ECU/server), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).

▽

△

Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to OperationOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in OperationOutput.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for TransferData following a previous RequestDownload. *	

⌋

8.13.1.1.3.2 Offer

[SWS_DM_00792] Definition of API function ara::diag::DownloadService::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	`class ara::diag::DownloadService`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.13.1.1.3.3 RequestDownload

[SWS_DM_00789] Definition of API function ara::diag::DownloadService::RequestDownload

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	`class ara::diag::DownloadService`	
Syntax:	`virtual ara::core::Future< void > RequestDownload (std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	dataFormatIdentifier	UDS dataFormat Identifier
	addressAndLengthFormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >	a Future, which either gets readied to void (for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RequestDownload.	

⌋

8.13.1.1.3.4 RequestDownloadExit

[SWS_DM_00791] Definition of API function ara::diag::DownloadService::RequestDownloadExit

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	`class ara::diag::DownloadService`	
Syntax:	`virtual ara::core::Future< OperationOutput > RequestDownloadExit (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	transferRequest ParameterRecord	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to OperationOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RequestTransferExit.	

⌋

8.13.1.1.3.5 StopOffer

[SWS_DM_00793] Definition of API function ara::diag::DownloadService::StopOffer

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.13.1.1.3.6 operator=

[SWS_DM_01647] Definition of API function ara::diag::DownloadService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	`class ara::diag::DownloadService`
Syntax:	`DownloadService & operator= (DownloadService &)=delete;`
Description:	DownloadService shall be a single not assignable instance.

⌋

8.13.2 Struct: OperationOutput

[SWS_DM_00785] Definition of API class ara::diag::DownloadService::OperationOutput

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	struct
Header file:	#include "ara/diag/download.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DownloadService`
Symbol:	OperationOutput
Syntax:	`struct OperationOutput {...};`
Description:	Response data of positive respone message.

⌋

8.13.2.1 Public Member Variables

8.13.2.1.1 responseData

[SWS_DM_00786] Definition of API variable ara::diag::DownloadService::OperationOutput::responseData

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	variable
Header file:	#include "ara/diag/download.h"
Scope:	`struct ara::diag::DownloadService::OperationOutput`
Symbol:	responseData
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> responseData;`
Description:	Content of positive respone message (without SID) Depending on the operation (e.g.: DownloadData, RequestDownloadExit) the expectation, what responseData shall contain (where it starts in the positive response) might differ. See doc of corresponding operation.

⌋

8.14 Header: ara/diag/dynamic_access_list_diag_service_builder.h

8.14.1 Class: DynamicAccessListDiagServiceBuilder

[SWS_DM_01166] Definition of API class ara::diag::DynamicAccessListDiagServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DynamicAccessListDiagServiceBuilder
Syntax:	`class DynamicAccessListDiagServiceBuilder final {...};`
Description:	Definition of the DynamicAccessListDiagServiceBuilder class, which is used by the application to build a DynamicAccessList.

⌋

8.14.1.1 Public Member Types

8.14.1.1.1 Type Alias: Byte

[SWS_DM_01167] Definition of API type ara::diag::DynamicAccessListDiagServiceBuilder::Byte

Upstream requirements: RS_Diag_04251

⌈

Kind:	type alias
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Symbol:	Byte
Syntax:	`using Byte = std::uint8_t;`
Description:	Type alias of a single diagnostic service pattern element in the DynamicAccessList.

⌋

8.14.1.1.2 Type Alias: ByteString

[SWS_DM_01168] Definition of API type ara::diag::DynamicAccessListDiagServiceBuilder::ByteString

Upstream requirements: RS_Diag_04251

⌈

Kind:	type alias
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Symbol:	ByteString
Syntax:	`using ByteString = ara::core::Span<Byte>;`
Description:	Type alias of a sequence of diagnostic service pattern elements in the DynamicAccessList.

⌋

8.14.1.2 Public Member Functions

8.14.1.2.1 Special Member Functions

8.14.1.2.1.1 Copy Constructor

[SWS_DM_01171] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagService Builder const &other)=delete;`
Description:	Copy constructor of DynamicAccessListDiagServiceBuilder cannot be used.

⌋

8.14.1.2.1.2 Move Constructor

[SWS_DM_01170] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagServiceBuilder &&other) noexcept;`	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of DynamicAccessListDiagServiceBuilder.	

⌋

8.14.1.2.1.3 Destructor

[SWS_DM_01174] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::~DynamicAccessListDiagServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`~DynamicAccessListDiagServiceBuilder () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DynamicAccessListDiagServiceBuilder .

⌋

8.14.1.2.2 Constructors

8.14.1.2.2.1 DynamicAccessListDiagServiceBuilder

[SWS_DM_01169] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`template <typename V> DynamicAccessListDiagServiceBuilder (V value, ara::core::Vector< std::uint8_t > &content) noexcept;`	
Parameters (in):	value	value(s) to be added to the DynamicAccessList
	content	Stack holder for serialized DynamicAccessList
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructor for DynamicAccessListDiagServiceBuilder.	

⌋

8.14.1.2.3 Member Functions

8.14.1.2.3.1 Add

[SWS_DM_01177] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`auto Add (ByteRange range) noexcept -> DynamicAccessListDiagServiceBuilder &;`	
Parameters (in):	range	The range of byte values to add to the DynamicAccessList. The range will be used to check for a match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	exception safe	

▽

△

Thread Safety:	not thread-safe
Description:	This function is used by the application to add a range of bytes to a DynamicAccessListPattern.

⌋

8.14.1.2.3.2 Add

[SWS_DM_01175] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`auto Add (Byte value) noexcept -> DynamicAccessListDiagServiceBuilder &;`	
Parameters (in):	value	The byte value to add to the DynamicAccessList. The value will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to add a single byte to a DynamicAccessListPattern.	

⌋

8.14.1.2.3.3 Add

[SWS_DM_01176] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`auto Add (ByteString values) noexcept -> DynamicAccessListDiagServiceBuilder &;`

▽

△

Parameters (in):	values	The byte sequence to add to the DynamicAccessList. The values will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiag ServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to add a string of bytes to a DynamicAccessListPattern.	

⌋

8.14.1.2.3.4 Any

[SWS_DM_01178] Definition of API function ara::diag::DynamicAccessListDiag ServiceBuilder::Any

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`auto Any (std::size_t numberOfBytesToIgnore) noexcept -> DynamicAccess ListDiagServiceBuilder &;`	
Parameters (in):	numberOfBytesToIgnore	The number of bytes to ignore
Return value:	DynamicAccessListDiag ServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to define a wildcard, i.e, to define a set of bytes that will be ignored in the DynamicAccessList pattern being created.	

⌋

8.14.1.2.3.5 Build

[SWS_DM_01181] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Build

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`void Build () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Finalizes building the DynamicAccessList. Must be called before destroying the ServiceBuilder object.

⌋

8.14.1.2.3.6 EndsWith

[SWS_DM_01180] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::EndsWith

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`void EndsWith (ByteRange range) noexcept;`	
Parameters (in):	range	The range of byte values to end the DynamicAccessList. The range will be used to check for a match during evaluation of the diagnostic service access rights
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to specify a closed range of bytes at the end of the DynamicAccessList pattern.	

⌋

8.14.1.2.3.7 EndsWith

[SWS_DM_01179] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::EndsWith

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`	
Syntax:	`void EndsWith (Byte value) noexcept;`	
Parameters (in):	value	The byte value to end the DynamicAccessList. The value will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the application to specify a single byte value at the end of the Dynamic AccessList pattern.	

⌋

8.14.1.2.3.8 operator=

[SWS_DM_01173] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`auto operator= (DynamicAccessListDiagServiceBuilder &&other) -> DynamicAccessListDiagServiceBuilder &=delete;`
Description:	Move assignment operator of DynamicAccessListDiagServiceBuilder.

⌋

8.14.1.2.3.9 operator=

[SWS_DM_01172] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Syntax:	`auto operator= (DynamicAccessListDiagServiceBuilder const &other) -> DynamicAccessListDiagServiceBuilder &=delete;`
Description:	Copy assignment operator of DynamicAccessListDiagServiceBuilder cannot be used.

⌋

8.14.2 Class: ByteRange

[SWS_DM_01182] Definition of API class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder`
Symbol:	ByteRange
Syntax:	`class ByteRange final {...};`
Description:	Represents a single byte value closed range. Type alias of a single diagnostic service pattern element in the DynamicAccessList

⌋

8.14.2.1 Public Member Functions

8.14.2.1.1 Special Member Functions

8.14.2.1.1.1 Move Constructor

[SWS_DM_01185] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`ByteRange (ByteRange &&other) noexcept;`	
DIRECTION NOT DEFINED	other	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of ByteRange.	

⌋

8.14.2.1.1.2 Copy Constructor

[SWS_DM_01184] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`ByteRange (ByteRange const &other) noexcept;`	
DIRECTION NOT DEFINED	other	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor of ByteRange.	

⌋

8.14.2.1.1.3 Destructor

[SWS_DM_01188] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::~ByteRange

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`
Syntax:	`~ByteRange () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ByteRange.

⌋

8.14.2.1.2 Constructors

8.14.2.1.2.1 ByteRange

[SWS_DM_01183] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`ByteRange (Byte min, Byte max) noexcept;`	
Parameters (in):	min	The minimum value to match
	max	The maximum value to match
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructs a value range.	

⌋

8.14.2.1.3 Member Functions

8.14.2.1.3.1 GetMax

[SWS_DM_01190] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`auto GetMax () const noexcept -> Byte;`	
Return value:	Byte	The the highest value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reports the highest value to match.	

⌋

8.14.2.1.3.2 GetMin

[SWS_DM_01189] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`auto GetMin () const noexcept -> Byte;`	
Return value:	Byte	The the lowest value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reports the lowest value to match.	

⌋

8.14.2.1.3.3 operator=

[SWS_DM_01186] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`auto operator= (ByteRange const &other) & noexcept -> ByteRange &;`	
DIRECTION NOT DEFINED	other	--
Return value:	ByteRange &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator of ByteRange .	

⌋

8.14.2.1.3.4 operator=

[SWS_DM_01187] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Scope:	`class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange`	
Syntax:	`auto operator= (ByteRange &&other) & noexcept -> ByteRange &;`	
DIRECTION NOT DEFINED	other	--
Return value:	ByteRange &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of ByteRange .	

⌋

8.15 Header: ara/diag/ecu_reset_request.h

8.15.1 Non-Member Types

8.15.1.1 Type Alias: ResetRequestType

[SWS_DM_01007] Definition of API type ara::diag::ResetRequestType

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	type alias
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`namespace ara::diag`
Symbol:	ResetRequestType
Syntax:	`using ResetRequestType = std::uint8_t;`
Description:	The type of the requested reset.

⌋

8.15.2 Global Variables

8.15.2.1 kCustomReset

[SWS_DM_01286] Definition of API variable ara::diag::kCustomReset

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`namespace ara::diag`
Symbol:	kCustomReset
Type:	`ResetRequestType`
Syntax:	`constexpr ResetRequestType kCustomReset = 3;`
Description:	kCustomReset

⌋

8.15.2.2 kHardReset

[SWS_DM_01284] Definition of API variable ara::diag::kHardReset

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`namespace ara::diag`
Symbol:	kHardReset
Type:	`ResetRequestType`
Syntax:	`constexpr ResetRequestType kHardReset = 1;`
Description:	HardReset.

⌋

8.15.2.3 kKeyOffOnReset

[SWS_DM_01285] Definition of API variable ara::diag::kKeyOffOnReset

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`namespace ara::diag`
Symbol:	kKeyOffOnReset
Type:	`ResetRequestType`
Syntax:	`constexpr ResetRequestType kKeyOffOnReset = 2;`
Description:	KeyOffOnReset.

⌋

8.15.2.4 kSoftReset

[SWS_DM_01283] Definition of API variable ara::diag::kSoftReset

Upstream requirements: RS_AP_00125

⌈

Kind:	variable
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`namespace ara::diag`
Symbol:	kSoftReset
Type:	`ResetRequestType`
Syntax:	`constexpr ResetRequestType kSoftReset = 0;`
Description:	SoftReset.

⌋

8.15.3 Class: EcuResetRequest

[SWS_DM_01009] Definition of API class ara::diag::EcuResetRequest

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	class
Header file:	#include "ara/diag/ecu_reset_request.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	EcuResetRequest
Syntax:	`class EcuResetRequest {...};`
Description:	Service EcuReset Request interface.

⌋

8.15.3.1 Public Member Functions

8.15.3.1.1 Special Member Functions

8.15.3.1.1.1 Move Constructor

[SWS_DM_01622] Definition of API function ara::diag::EcuResetRequest::EcuResetRequest

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`
Syntax:	`EcuResetRequest (EcuResetRequest &&) noexcept=delete;`
Description:	Move constructor of EcuResetRequest.

⌋

8.15.3.1.1.2 Move Assignment Operator

[SWS_DM_01620] Definition of API function ara::diag::EcuResetRequest::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`
Syntax:	`EcuResetRequest & operator= (EcuResetRequest &&)=delete;`
Description:	Move assignment operator of EcuResetRequest.

⌋

8.15.3.1.1.3 Destructor

[SWS_DM_01011] Definition of API function ara::diag::EcuResetRequest::~EcuResetRequest

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`
Syntax:	`virtual ~EcuResetRequest () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of EcuResetRequest .

⌋

8.15.3.1.2 Constructors

8.15.3.1.2.1 EcuResetRequest

[SWS_DM_01010] Definition of API function ara::diag::EcuResetRequest::EcuResetRequest

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	`class ara::diag::EcuResetRequest`	
Syntax:	`explicit EcuResetRequest (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticEcuResetInterface` needs to be refernced by a `DiagnosticServiceGenericMapping`.

▽

△

	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of EcuResetRequest.	

⌋

8.15.3.1.2.2 EcuResetRequest

[SWS_DM_01621] Definition of API function ara::diag::EcuResetRequest::EcuResetRequest

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`
Syntax:	`EcuResetRequest (EcuResetRequest &)=delete;`
Description:	EcuResetRequest shall be a single not copy-able instance.

⌋

8.15.3.1.3 Member Functions

8.15.3.1.3.1 EnableRapidShutdown

[SWS_DM_01012] Definition of API function ara::diag::EcuResetRequest::EnableRapidShutdown

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`

▽

△

Syntax:	`virtual ara::core::Future< void > EnableRapidShutdown (bool enable, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	enable	when enable is set to true the rapid shutdown will be enabled, setting enable to false will disable rapid shutdown
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	interface for subFunction En-/DisableRapidShutdown	

⌋

8.15.3.1.3.2 ExecuteReset

[SWS_DM_01014] Definition of API function ara::diag::EcuResetRequest::ExecuteReset

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	`class ara::diag::EcuResetRequest`	
Syntax:	`virtual ara::core::Future< void > ExecuteReset (const MetaInfo &meta Info) noexcept=0;`	
DIRECTION NOT DEFINED	metaInfo	--
Return value:	ara::core::Future< void >	Return nothing or an error. In case the parameter DiagnosticEcu ResetClass.respondToReset is either not present or present and set to respondBeforeReset, a given error has no effect
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	process which is instantiating this interface, has to execute the requested reset.	

⌋

8.15.3.1.3.3 Offer

[SWS_DM_01016] Definition of API function ara::diag::EcuResetRequest::Offer

Upstream requirements: RS_AP_00139, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	`class ara::diag::EcuResetRequest`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.15.3.1.3.4 RequestReset

[SWS_DM_01013] Definition of API function ara::diag::EcuResetRequest::RequestReset

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	`class ara::diag::EcuResetRequest`	
Syntax:	`virtual ara::core::Future< void > RequestReset (ResetRequestType resetType, ara::core::Optional< std::uint8_t > id, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	resetType	Type of the requested reset.
	id	id of the custom reset type. Will only be evaluated when resetType is "custom"
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< void >	--
Exception Safety:	exception safe	

▽

△

Thread Safety:	not thread-safe	
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for any EcuReset subFunction, except En-/DisableRapidShutdown. process, which is instantiating this interface, needs to evalute carefully if the request to restart parts or the whole machine. Once the request to reset is accepted, the process, which is instantiating this interface, has to rely on this decision for the ExecuteReset() trigger	

⌋

8.15.3.1.3.5 StopOffer

[SWS_DM_01017] Definition of API function ara::diag::EcuResetRequest::StopOffer

Upstream requirements: RS_Diag_04169, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.15.3.1.3.6 operator=

[SWS_DM_01619] Definition of API function ara::diag::EcuResetRequest::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	`class ara::diag::EcuResetRequest`

▽

△

Syntax:	`EcuResetRequest & operator= (EcuResetRequest &)=delete;`
Description:	EcuResetRequest shall be a single not assignable instance.

⌋

8.16 Header: ara/diag/external_authentication.h

8.16.1 Class: ExternalAuthentication

[SWS_DM_01191] Definition of API class ara::diag::ExternalAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/external_authentication.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	ExternalAuthentication
Syntax:	`class ExternalAuthentication final {...};`
Description:	Definition of the ExternalAuthentication class, which is used by the application to receive an instance of the ClientAuthentication Class relevant to the specific client.

⌋

8.16.1.1 Public Member Types

8.16.1.1.1 Type Alias: Address

[SWS_DM_01192] Definition of API type ara::diag::ExternalAuthentication::Address

Upstream requirements: RS_Diag_04251

⌈

Kind:	type alias
Header file:	#include "ara/diag/external_authentication.h"
Scope:	`class ara::diag::ExternalAuthentication`
Symbol:	Address

▽

△

Syntax:	`using Address = std::uint16_t;`
Description:	Alias for tester address.

⌋

8.16.1.2 Public Member Functions

8.16.1.2.1 Special Member Functions

8.16.1.2.1.1 Copy Constructor

[SWS_DM_01196] Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/external_authentication.h"
Scope:	`class ara::diag::ExternalAuthentication`
Syntax:	`ExternalAuthentication (ExternalAuthentication const &other)=delete;`
Description:	Copy constructor of ExternalAuthentication.

⌋

8.16.1.2.1.2 Move Constructor

[SWS_DM_01194] Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`ExternalAuthentication (ExternalAuthentication &&other) noexcept=default;`	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	

▽

△

Thread Safety:	implementation defined
Description:	Move constructor of ExternalAuthentication.

⌋

8.16.1.2.1.3 Destructor

[SWS_DM_01198] Definition of API function ara::diag::ExternalAuthentication::~ExternalAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/external_authentication.h"
Scope:	`class ara::diag::ExternalAuthentication`
Syntax:	`~ExternalAuthentication () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticServiceDynamicAccessList .

⌋

8.16.1.2.2 Constructors

8.16.1.2.2.1 ExternalAuthentication

[SWS_DM_01193] Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`explicit ExternalAuthentication (ara::core::InstanceSpecifier instanceSpecifier) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a DiagnosticAuthentication service instance in the manifest

▽

△

Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticExternalAuthenticationPortMapping` needs to be typed by a `DiagnosticExternalAuthenticationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructs the port for authentication of diagnostic clients.	

⌋

8.16.1.2.3 Member Functions

8.16.1.2.3.1 Get

[SWS_DM_01200] Definition of API function ara::diag::ExternalAuthentication::Get

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`ara::core::Result< ClientAuthentication > Get (Address sourceAddress) noexcept;`	
Parameters (in):	sourceAddress	The source address of the client
Return value:	ara::core::Result< ClientAuthentication >	The associated diagnostic client authentication object or error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.

▽

△

Description:	This function is used by the application to get the ClientAuthentication Instance that is handling the Authentication State of the Client corresponding to the Address.

⌋

8.16.1.2.3.2 Get

[SWS_DM_01199] Definition of API function ara::diag::ExternalAuthentication::Get

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`ara::core::Result< ClientAuthentication > Get (const MetaInfo &meta Info) noexcept;`	
Parameters (in):	metaInfo	The meta information of a diagnostic service port
Return value:	ara::core::Result< Client Authentication >	The diagnostic client associated authentication object or error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	This function is used by the application to get the ClientAuthentication Instance that is handling the Authentication State of the Client corresponding to the MetaInfo.	

⌋

8.16.1.2.3.3 GetAll

[SWS_DM_01201] Definition of API function ara::diag::ExternalAuthentication::GetAll

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`ara::core::Vector< ClientAuthentication > GetAll () noexcept;`	
Return value:	ara::core::Vector< Client Authentication >	The list of all diagnostic client associated authentication objects or empty list if none available.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	This function is used by the application to get all the ClientAuthentication Instances that are currently handled by the DM.	

⌋

8.16.1.2.3.4 operator=

[SWS_DM_01197] Definition of API function ara::diag::ExternalAuthentication::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/external_authentication.h"
Scope:	`class ara::diag::ExternalAuthentication`
Syntax:	`auto operator= (ExternalAuthentication const &other) -> ExternalAuthentication &=delete;`
Description:	Copy assignment operator of ExternalAuthentication.

⌋

8.16.1.2.3.5 operator=

[SWS_DM_01195] Definition of API function ara::diag::ExternalAuthentication::operator=

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	`class ara::diag::ExternalAuthentication`	
Syntax:	`auto operator= (ExternalAuthentication &&other) & noexcept -> ExternalAuthentication & =default;`	
Parameters (in):	other	Object to move-assign from.
Return value:	ExternalAuthentication & =default	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of ExternalAuthentication.	

⌋

8.17 Header: ara/diag/file_transfer.h

8.17.1 Class: FileTransferService

[SWS_DM_01320] Definition of API class ara::diag::FileTransferService

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/file_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	FileTransferService
Syntax:	`class FileTransferService {...};`
Description:	File Transfer service interface .

⌋

8.17.1.1 Public Member Types

8.17.1.1.1 Enumeration: WriteFileMode

[SWS_DM_01324] Definition of API enum ara::diag::FileTransferService::WriteFileMode

Upstream requirements: RS_Diag_04135

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/file_transfer.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`class ara::diag::FileTransferService`	
Symbol:	WriteFileMode	
Underlying type:	ara::core::Byte	
Syntax:	`enum class WriteFileMode : ara::core::Byte {...};`	
Values:	kAdd	The file shall be added only if not existing.
	kReplace	The file shall be added in any case, if already exists, will be replaced
Description:	Determines the write file operation mode .	

⌋

8.17.1.2 Public Member Functions

8.17.1.2.1 Special Member Functions

8.17.1.2.1.1 Destructor

[SWS_DM_01326] Definition of API function ara::diag::FileTransferService::~FileTransferService

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`class ara::diag::FileTransferService`
Syntax:	`virtual ~FileTransferService () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class FileTransferService.

⌋

8.17.1.2.2 Constructors

8.17.1.2.2.1 FileTransferService

[SWS_DM_01325] Definition of API function ara::diag::FileTransferService::FileTransferService

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`FileTransferService (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticRequestFileTransferInterface` needs to be referenced by a `DiagnosticServiceGenericMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for FileTransferService (inherited)	

⌋

8.17.1.2.3 Member Functions

8.17.1.2.3.1 DeleteFile

[SWS_DM_01335] Definition of API function ara::diag::FileTransferService::DeleteFile

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`virtual auto DeleteFile (ara::core::String fileName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< void >=0;`	
Parameters (in):	fileName	Path including name of the file to read
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >=0	A future with either a void data result or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for RequestFileTransfer with ModeOfOperation DeleteFile. This method is the complete operation of deleting a file	

⌋

8.17.1.2.3.2 Offer

[SWS_DM_01336] Definition of API function ara::diag::FileTransferService::Offer

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`class ara::diag::FileTransferService`
Syntax:	`auto Offer () noexcept -> ara::core::Result< void >;`

▽

△

Return value:	ara::core::Result< void >	Positive result if service was offered, error if offer failed
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.17.1.2.3.3 RequestReadDirectory

[SWS_DM_01332] Definition of API function ara::diag::FileTransferService::RequestReadDirectory

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`virtual auto RequestReadDirectory (ara::core::String directoryName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< std::tuple< DataTransferReadSession, std::uint64_t > >=0;`	
Parameters (in):	directoryName	Path including name of the directory to read
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< std::tuple< DataTransfer ReadSession, std::uint64_t > >=0	A future with either the access strategy to be used during reading and the directory information size to be reported to the UDS client for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for RequestFileTransfer with ModeOfOperation ReadDir.	

⌋

8.17.1.2.3.4 RequestReadFile

[SWS_DM_01331] Definition of API function ara::diag::FileTransferService::RequestReadFile

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`virtual auto RequestReadFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< std::tuple< DataTransferReadSession, FileSizes > >=0;`	
Parameters (in):	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< std::tuple< DataTransfer ReadSession, FileSizes > >=0	A future with either the access strategy to be used during reading and the file sizes to be reported to the UDS client for a positive response message) or an UDS NRC value (for an negative response message) uncompressed size: If set to 0 - the parameter will not be reported to the diagnostic client compressed size: If set to 0 - the parameter will not be reported to the diagnostic client. If no compression is used, the value shall be zero.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for RequestFileTransfer with ModeOfOperation ReadFile.	

⌋

8.17.1.2.3.5 RequestResumeWriteFile

[SWS_DM_01334] Definition of API function ara::diag::FileTransferService::RequestResumeWriteFile

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`virtual auto RequestResumeWriteFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes fileSizes, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< std::tuple< DataTransferWriteSession, std::uint64_t > >=0;`	
Parameters (in):	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	fileSizes	The compressed/uncompressed files sizes
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< std::tuple< DataTransfer WriteSession, std::uint64_t > >=0	A future with either an instance of the file writing session and the byte position the client shall start resuming from or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for RequestFileTransfer with ModeOfOperation ResumeFile.	

⌋

8.17.1.2.3.6 RequestWriteFile

[SWS_DM_01333] Definition of API function ara::diag::FileTransferService::RequestWriteFile

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`virtual auto RequestWriteFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes fileSizes, WriteFile Mode mode, MetaInfo const &metaInfo, CancellationHandler cancellation Handler) noexcept -> ara::core::Future< DataTransferWriteSession >=0;`	
Parameters (in):	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	fileSizes	The compressed/uncompressed files sizes
	mode	The file replacement mode
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< Data TransferWriteSession >=0	A future with either an instance of the file writing session or an UDS NRC value (for an negative response message) A future with either an instance of the file writing session or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for RequestFileTransfer with ModeOfOperation WriteFile.	

⌋

8.17.1.2.3.7 StopOffer

[SWS_DM_01337] Definition of API function ara::diag::FileTransferService::StopOffer

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`class ara::diag::FileTransferService`
Syntax:	`void StopOffer () const noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

⌋

8.17.1.3 Protected Member Functions

8.17.1.3.1 Special Member Functions

8.17.1.3.1.1 Copy Constructor

[SWS_DM_01327] Definition of API function ara::diag::FileTransferService::FileTransferService

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`class ara::diag::FileTransferService`
Syntax:	`FileTransferService (FileTransferService const &)=delete;`
Description:	FileTransferService shall be a single not copy-able instance. .
Visibility:	protected

⌋

8.17.1.3.1.2 Move Constructor

[SWS_DM_01328] Definition of API function ara::diag::FileTransferService::FileTransferService

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`FileTransferService (FileTransferService &&other) noexcept;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of FileTransferService.	
Visibility:	protected	

⌋

8.17.1.3.2 Member Functions

8.17.1.3.2.1 operator=

[SWS_DM_01329] Definition of API function ara::diag::FileTransferService::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`class ara::diag::FileTransferService`
Syntax:	`auto operator= (FileTransferService const &) -> FileTransferService &=delete;`
Description:	FileTransferService shall be a single not assignable instance.
Visibility:	protected

⌋

8.17.1.3.2.2 operator=

[SWS_DM_01330] Definition of API function ara::diag::FileTransferService::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	`class ara::diag::FileTransferService`	
Syntax:	`auto operator= (FileTransferService &&other) & noexcept -> FileTransferService &;`	
Parameters (out):	other	The other object
Return value:	FileTransferService &	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of FileTransferService.	
Visibility:	protected	

⌋

8.17.2 Struct: FileSizes

[SWS_DM_01321] Definition of API class ara::diag::FileTransferService::FileSizes

Upstream requirements: RS_Diag_04135

⌈

Kind:	struct
Header file:	#include "ara/diag/file_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::FileTransferService`
Symbol:	FileSizes
Syntax:	`struct FileSizes {...};`
Description:	Definition of total file sizes .

⌋

8.17.2.1 Public Member Variables

8.17.2.1.1 compressed_size

[SWS_DM_01323] Definition of API variable ara::diag::FileTransferService::FileSizes::compressed_size

Upstream requirements: RS_Diag_04135

⌈

Kind:	variable
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`struct ara::diag::FileTransferService::FileSizes`
Symbol:	compressed_size
Type:	`std::uint64_t`
Syntax:	`std::uint64_t compressed_size;`
Description:	Specifies the compressed file size in bytes .

⌋

8.17.2.1.2 uncompressed_size

[SWS_DM_01322] Definition of API variable ara::diag::FileTransferService::FileSizes::uncompressed_size

Upstream requirements: RS_Diag_04135

⌈

Kind:	variable
Header file:	#include "ara/diag/file_transfer.h"
Scope:	`struct ara::diag::FileTransferService::FileSizes`
Symbol:	uncompressed_size
Type:	`std::uint64_t`
Syntax:	`std::uint64_t uncompressed_size;`
Description:	Specifies the uncompressed file size in bytes .

⌋

8.18 Header: ara/diag/generic_data_identifier.h

8.18.1 Class: GenericDataIdentifier

[SWS_DM_00607] Definition of API class ara::diag::GenericDataIdentifier

Upstream requirements: RS_Diag_04169

⌈

Kind:	class
Header file:	#include "ara/diag/generic_data_identifier.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	GenericDataIdentifier
Syntax:	`class GenericDataIdentifier {...};`
Description:	Generic DataIdentifier interface.

⌋

8.18.1.1 Public Member Functions

8.18.1.1.1 Special Member Functions

8.18.1.1.1.1 Move Constructor

[SWS_DM_01654] Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Syntax:	`GenericDataIdentifier (GenericDataIdentifier &&) noexcept=delete;`
Description:	Move constructor of GenericDataIdentifier.

⌋

8.18.1.1.1.2 Move Assignment Operator

[SWS_DM_01652] Definition of API function ara::diag::GenericDataIdentifier::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Syntax:	`GenericDataIdentifier & operator= (GenericDataIdentifier &&)=delete;`
Description:	Move assignment operator of GenericDataIdentifier.

⌋

8.18.1.1.1.3 Destructor

[SWS_DM_00635] Definition of API function ara::diag::GenericDataIdentifier::~GenericDataIdentifier

Upstream requirements: RS_AP_00134, RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Syntax:	`virtual ~GenericDataIdentifier () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class GenericDataIdentifier .

⌋

8.18.1.1.2 Constructors

8.18.1.1.2.1 GenericDataIdentifier

[SWS_DM_00634] Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier

Upstream requirements: RS_AP_00137, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_data_identifier.h"	
Scope:	`class ara::diag::GenericDataIdentifier`	
Syntax:	`explicit GenericDataIdentifier (const ara::core::InstanceSpecifier &specifier, DataIdentifierConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an `DiagnosticDataIdentifierGenericInterface`
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticDataIdentifierGenericInterface` needs to be referenced by a `DiagnosticDataPortMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Class for an GenericDataIdentifier.	

⌋

8.18.1.1.2.2 GenericDataIdentifier

[SWS_DM_01653] Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Syntax:	`GenericDataIdentifier (GenericDataIdentifier &)=delete;`
Description:	GenericDataIdentifier shall be a single not copy-able instance.

⌋

8.18.1.1.3 Member Functions

8.18.1.1.3.1 Offer

[SWS_DM_00638] Definition of API function ara::diag::GenericDataIdentifier::Offer

Upstream requirements: RS_AP_00139, RS_AP_00119, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_data_identifier.h"	
Scope:	`class ara::diag::GenericDataIdentifier`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.18.1.1.3.2 Read

[SWS_DM_00636] Definition of API function ara::diag::GenericDataIdentifier::Read

Upstream requirements: RS_AP_00138, RS_AP_00119, RS_Diag_04169, RS_Diag_04170, RS_Diag_04172

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_data_identifier.h"	
Scope:	`class ara::diag::GenericDataIdentifier`	
Syntax:	`virtual ara::core::Future< OperationOutput > Read (std::uint16_t data Identifier, const MetaInfo &metaInfo, CancellationHandler cancellation Handler) noexcept=0;`	
Parameters (in):	dataIdentifier	the corresponding DataIdentifier
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for ReadDataByIdentifier request for this DiagnosticDataIdentifier.	

⌋

8.18.1.1.3.3 StopOffer

[SWS_DM_00639] Definition of API function ara::diag::GenericDataIdentifier::StopOffer

Upstream requirements: RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`

▽

△

Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.18.1.1.3.4 Write

[SWS_DM_00637] Definition of API function ara::diag::GenericDataIdentifier::Write

Upstream requirements: RS_AP_00138, RS_AP_00119, RS_Diag_04169, RS_Diag_04170

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_data_identifier.h"	
Scope:	`class ara::diag::GenericDataIdentifier`	
Syntax:	`virtual ara::core::Future< void > Write (std::uint16_t dataIdentifier, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept;`	
Parameters (in):	dataIdentifier	the corresponding DataIdentifier
	requestData	Content of request message (without DataIdentifier), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for WriteDataByIdentifier request for this DiagnosticDataIdentifier.	

⌋

8.18.1.1.3.5 operator=

[SWS_DM_01651] Definition of API function ara::diag::GenericDataIdentifier::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Syntax:	`GenericDataIdentifier & operator= (GenericDataIdentifier &)=delete;`
Description:	GenericDataIdentifier shall be a single not assignable instance.

⌋

8.18.2 Struct: OperationOutput

[SWS_DM_00641] Definition of API class ara::diag::GenericDataIdentifier::OperationOutput

Upstream requirements: RS_Diag_04169, RS_Diag_04172

⌈

Kind:	struct
Header file:	#include "ara/diag/generic_data_identifier.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::GenericDataIdentifier`
Symbol:	OperationOutput
Syntax:	`struct OperationOutput {...};`
Description:	Response data of positive respone message.

⌋

8.18.2.1 Public Member Variables

8.18.2.1.1 responseData

[SWS_DM_00631] Definition of API variable ara::diag::GenericDataIdentifier::OperationOutput::responseData

Upstream requirements: RS_AP_00137, RS_Diag_04169

⌈

Kind:	variable
Header file:	#include "ara/diag/generic_data_identifier.h"
Scope:	`struct ara::diag::GenericDataIdentifier::OperationOutput`
Symbol:	responseData
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> responseData;`
Description:	Content of positive respone message (without DataIdentifier)

⌋

8.19 Header: ara/diag/generic_routine.h

8.19.1 Class: GenericRoutine

[SWS_DM_00605] Definition of API class ara::diag::GenericRoutine

Upstream requirements: RS_Diag_04169, RS_Diag_04224

⌈

Kind:	class
Header file:	#include "ara/diag/generic_routine.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	GenericRoutine
Syntax:	`class GenericRoutine {...};`
Description:	Generic Routine interface.

⌋

8.19.1.1 Public Member Functions

8.19.1.1.1 Special Member Functions

8.19.1.1.1.1 Move Constructor

[SWS_DM_01638] Definition of API function ara::diag::GenericRoutine::GenericRoutine

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`class ara::diag::GenericRoutine`
Syntax:	`GenericRoutine (GenericRoutine &&) noexcept=delete;`
Description:	Move constructor of GenericRoutine.

⌋

8.19.1.1.1.2 Move Assignment Operator

[SWS_DM_01636] Definition of API function ara::diag::GenericRoutine::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`class ara::diag::GenericRoutine`
Syntax:	`GenericRoutine & operator= (GenericRoutine &&)=delete;`
Description:	Move assignment operator of GenericRoutine.

⌋

8.19.1.1.1.3 Destructor

[SWS_DM_00553] Definition of API function ara::diag::GenericRoutine::~Generic Routine

Upstream requirements: RS_AP_00134, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`class ara::diag::GenericRoutine`
Syntax:	`virtual ~GenericRoutine () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class GenericRoutine .

⌋

8.19.1.1.2 Constructors

8.19.1.1.2.1 GenericRoutine

[SWS_DM_00552] Definition of API function ara::diag::GenericRoutine::Generic Routine

Upstream requirements: RS_AP_00137, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	`class ara::diag::GenericRoutine`	
Syntax:	`explicit GenericRoutine (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticRoutine GenericInterface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticRoutineGenericInterface needs to be referenced by a DiagnosticServiceGenericMapping.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Class for an GenericRoutine.	

⌋

8.19.1.1.2.2 GenericRoutine

[SWS_DM_01637] Definition of API function ara::diag::GenericRoutine::GenericRoutine

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	class ara::diag::GenericRoutine
Syntax:	GenericRoutine (GenericRoutine &)=delete;
Description:	GenericRoutine shall be a single not copy-able instance.

⌋

8.19.1.1.3 Member Functions

8.19.1.1.3.1 Offer

[SWS_DM_00557] Definition of API function ara::diag::GenericRoutine::Offer

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Main_01002, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	`class ara::diag::GenericRoutine`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.19.1.1.3.2 RequestResults

[SWS_DM_00556] Definition of API function ara::diag::GenericRoutine::Request Results

Upstream requirements: RS_AP_00119, RS_Diag_04169, RS_Diag_04170, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	`class ara::diag::GenericRoutine`	
Syntax:	`virtual ara::core::Future< OperationOutput > RequestResults (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept;`	
Parameters (in):	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information

▽

△

	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RoutineControl with SubFunction RequestResults request for this DiagnosticRoutine Identifier.	

⌋

8.19.1.1.3.3 Start

[SWS_DM_00554] Definition of API function ara::diag::GenericRoutine::Start

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04169, RS_Diag_04170, RS_-Diag_04224

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	`class ara::diag::GenericRoutine`	
Syntax:	`virtual ara::core::Future< OperationOutput > Start (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	

▽

△

Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RoutineControl with SubFunction Start request for this DiagnosticRoutineIdentifier.	

⌋

8.19.1.1.3.4 Stop

[SWS_DM_00555] Definition of API function ara::diag::GenericRoutine::Stop

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04169, RS_Diag_04170, RS_-Diag_04224

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	`class ara::diag::GenericRoutine`	
Syntax:	`virtual ara::core::Future< OperationOutput > Stop (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept;`	
Parameters (in):	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RoutineControl with SubFunction Stop request for this DiagnosticRoutineIdentifier.	

⌋

8.19.1.1.3.5 StopOffer

[SWS_DM_00558] Definition of API function ara::diag::GenericRoutine::StopOffer

Upstream requirements: RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`class ara::diag::GenericRoutine`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.19.1.1.3.6 operator=

[SWS_DM_01635] Definition of API function ara::diag::GenericRoutine::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`class ara::diag::GenericRoutine`
Syntax:	`GenericRoutine & operator= (GenericRoutine &)=delete;`
Description:	GenericRoutine shall be a single not assignable instance.

⌋

8.19.2 Struct: OperationOutput

[SWS_DM_00551] Definition of API class ara::diag::GenericRoutine::OperationOutput

Upstream requirements: RS_Diag_04169, RS_Diag_04224

⌈

Kind:	struct
Header file:	#include "ara/diag/generic_routine.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::GenericRoutine`
Symbol:	OperationOutput
Syntax:	`struct OperationOutput {...};`
Description:	Response data of positive respone message.

⌋

8.19.2.1 Public Member Variables

8.19.2.1.1 responseData

[SWS_DM_00633] Definition of API variable ara::diag::GenericRoutine::OperationOutput::responseData

Upstream requirements: RS_AP_00137, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	variable
Header file:	#include "ara/diag/generic_routine.h"
Scope:	`struct ara::diag::GenericRoutine::OperationOutput`
Symbol:	responseData
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> responseData;`
Description:	Content of positive respone message (without RoutineIdentifier)

⌋

8.20 Header: ara/diag/generic_uds_service.h

8.20.1 Class: GenericUDSService

[SWS_DM_00602] Definition of API class ara::diag::GenericUDSService

Upstream requirements: RS_Diag_04169

⌈

Kind:	class
Header file:	#include "ara/diag/generic_uds_service.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	GenericUDSService
Syntax:	`class GenericUDSService {...};`
Description:	Generic UDS interface.

⌋

8.20.1.1 Public Member Functions

8.20.1.1.1 Special Member Functions

8.20.1.1.1.1 Move Constructor

[SWS_DM_01658] Definition of API function ara::diag::GenericUDSService::GenericUDSService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`
Syntax:	`GenericUDSService (GenericUDSService &&) noexcept=delete;`
Description:	Move constructor of GenericUDSService.

⌋

8.20.1.1.1.2 Move Assignment Operator

[SWS_DM_01656] Definition of API function ara::diag::GenericUDSService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`
Syntax:	`GenericUDSService & operator= (GenericUDSService &&)=delete;`
Description:	Move assignment operator of GenericUDSService.

⌋

8.20.1.1.1.3 Destructor

[SWS_DM_00584] Definition of API function ara::diag::GenericUDSService::~GenericUDSService

Upstream requirements: RS_AP_00134, RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`
Syntax:	`virtual ~GenericUDSService () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of GenericUDSService .

⌋

8.20.1.1.2 Constructors

8.20.1.1.2.1 GenericUDSService

[SWS_DM_00616] Definition of API function ara::diag::GenericUDSService::GenericUDSService

Upstream requirements: RS_AP_00137, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_uds_service.h"	
Scope:	`class ara::diag::GenericUDSService`	
Syntax:	`explicit GenericUDSService (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticGenericUdsInterface` needs to be referenced by a `DiagnosticServiceGenericMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of GenericUDSService.	

⌋

8.20.1.1.2.2 GenericUDSService

[SWS_DM_01657] Definition of API function ara::diag::GenericUDSService::GenericUDSService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`
Syntax:	`GenericUDSService (GenericUDSService &)=delete;`
Description:	GenericUDSService shall be a single not copy-able instance.

⌋

8.20.1.1.3 Member Functions

8.20.1.1.3.1 HandleMessage

[SWS_DM_00618] Definition of API function ara::diag::GenericUDSService::HandleMessage

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04169, RS_Diag_04170, RS_Diag_04172, RS_Diag_04173

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_uds_service.h"	
Scope:	`class ara::diag::GenericUDSService`	
Syntax:	`virtual ara::core::Future< OperationOutput > HandleMessage (std::uint8_t sid, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	sid	Diagnostic Request Service Identifier.
	requestData	Diagnostic request data (starting after SID), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< OperationOutput >	a Result with either a OperationOutput or an error

▽

△

Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for any request messsage.	

⌋

8.20.1.1.3.2 Offer

[SWS_DM_00619] Definition of API function ara::diag::GenericUDSService::Offer

Upstream requirements: RS_AP_00139, RS_AP_00119, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/generic_uds_service.h"	
Scope:	`class ara::diag::GenericUDSService`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.20.1.1.3.3 StopOffer

[SWS_DM_00620] Definition of API function ara::diag::GenericUDSService::Stop Offer

Upstream requirements: RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`

▽

△

Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.20.1.1.3.4 operator=

[SWS_DM_01655] Definition of API function ara::diag::GenericUDSService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`class ara::diag::GenericUDSService`
Syntax:	`GenericUDSService & operator= (GenericUDSService &)=delete;`
Description:	GenericUDSService shall be a single not assignable instance.

⌋

8.20.2 Struct: OperationOutput

[SWS_DM_00578] Definition of API class ara::diag::GenericUDSService::OperationOutput

Upstream requirements: RS_Diag_04169, RS_Diag_04172

⌈

Kind:	struct
Header file:	#include "ara/diag/generic_uds_service.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::GenericUDSService`
Symbol:	OperationOutput
Syntax:	`struct OperationOutput {...};`
Description:	Response data of positive respone message.

⌋

8.20.2.1 Public Member Variables

8.20.2.1.1 responseData

[SWS_DM_00632] Definition of API variable ara::diag::GenericUDSService::OperationOutput::responseData

Upstream requirements: RS_Diag_04169, RS_Diag_04172

⌈

Kind:	variable
Header file:	#include "ara/diag/generic_uds_service.h"
Scope:	`struct ara::diag::GenericUDSService::OperationOutput`
Symbol:	responseData
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> responseData;`
Description:	Content of positive respone message (without SID)

⌋

8.21 Header: ara/diag/meta_info.h

8.21.1 Class: MetaInfo

[SWS_DM_00971] Definition of API class ara::diag::MetaInfo

Upstream requirements: RS_Diag_04170

⌈

Kind:	class
Header file:	#include "ara/diag/meta_info.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	MetaInfo
Syntax:	`class MetaInfo final {...};`
Description:	Metainfo interface.

⌋

8.21.1.1 Public Member Types

8.21.1.1.1 Type Alias: Context

[SWS_DM_00977] Definition of API type ara::diag::MetaInfo::Context

Upstream requirements: RS_Diag_04170

⌈

Kind:	type alias
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Symbol:	Context
Syntax:	`using Context = std::uint32_t;`
Description:	Definition of possible call context .

⌋

8.21.1.2 Public Member Variables

8.21.1.2.1 kDiagnosticCommunication

[SWS_DM_01342] Definition of API variable ara::diag::MetaInfo::kDiagnostic Communication

Upstream requirements: RS_Diag_04170

⌈

Kind:	variable
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Symbol:	kDiagnosticCommunication
Type:	`MetaInfo::Context`
Syntax:	`MetaInfo::Context kDiagnosticCommunication = 0x01;`
Description:	Service request in DCM context.

⌋

8.21.1.2.2 kDoIP

[SWS_DM_01344] Definition of API variable ara::diag::MetaInfo::kDoIP

Upstream requirements: RS_Diag_04170

⌈

Kind:	variable
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Symbol:	kDoIP
Type:	`MetaInfo::Context`
Syntax:	`MetaInfo::Context kDoIP = 0x03;`
Description:	For reading VIN.

⌋

8.21.1.2.3 kFaultMemory

[SWS_DM_01343] Definition of API variable ara::diag::MetaInfo::kFaultMemory

Upstream requirements: RS_Diag_04170

⌈

Kind:	variable
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Symbol:	kFaultMemory
Type:	`MetaInfo::Context`
Syntax:	`MetaInfo::Context kFaultMemory = 0x02;`
Description:	For DIDs in Snapshots.

⌋

8.21.1.2.4 kSovd

[SWS_DM_01818] Definition of API variable ara::diag::MetaInfo::kSovd

Upstream requirements: RS_Diag_04170

⌈

Kind:	variable
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Symbol:	kSovd
Type:	`MetaInfo::Context`
Syntax:	`MetaInfo::Context kSovd = 0x04;`
Description:	Service request in SOVD context.

⌋

8.21.1.3 Public Member Functions

8.21.1.3.1 Special Member Functions

8.21.1.3.1.1 Copy Constructor

[SWS_DM_00973] Definition of API function ara::diag::MetaInfo::MetaInfo

Upstream requirements: RS_Diag_04170, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Syntax:	`MetaInfo (const MetaInfo &)=delete;`
Description:	Copy Constructor of MetaInfo cannot be used.

⌋

8.21.1.3.1.2 Default Constructor

[SWS_DM_00972] Definition of API function ara::diag::MetaInfo::MetaInfo

Upstream requirements: RS_Diag_04170, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Syntax:	`MetaInfo () noexcept=delete;`
Description:	Constructor of MetaInfo cannot be used.

⌋

8.21.1.3.1.3 Move Constructor

[SWS_DM_00974] Definition of API function ara::diag::MetaInfo::MetaInfo

Upstream requirements: RS_Diag_04170, RS_AP_00145

⌈

Kind:	function	
Header file:	#include "ara/diag/meta_info.h"	
Scope:	`class ara::diag::MetaInfo`	
Syntax:	`MetaInfo (MetaInfo &&obj) noexcept;`	
Parameters (in):	obj	object to be moved
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move Constructor of MetaInfo.	

⌋

8.21.1.3.1.4 Copy Assignment Operator

[SWS_DM_00975] Definition of API function ara::diag::MetaInfo::operator=

Upstream requirements: RS_Diag_04170, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Syntax:	`MetaInfo & operator= (const MetaInfo &)=delete;`
Description:	Copy Assignment Operator of MetaInfo cannot be used.

⌋

8.21.1.3.1.5 Move Assignment Operator

[SWS_DM_00976] Definition of API function ara::diag::MetaInfo::operator=

Upstream requirements: RS_Diag_04170, RS_AP_00145

⌈

Kind:	function	
Header file:	#include "ara/diag/meta_info.h"	
Scope:	`class ara::diag::MetaInfo`	
Syntax:	`MetaInfo & operator= (MetaInfo &&other) & noexcept;`	
Parameters (in):	other	MetaInfo instance
Return value:	MetaInfo &	Reference to the current object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move Assignment Operator of MetaInfo.	

⌋

8.21.1.3.1.6 Destructor

[SWS_DM_00980] Definition of API function ara::diag::MetaInfo::~MetaInfo

Upstream requirements: RS_Diag_04170, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/diag/meta_info.h"
Scope:	`class ara::diag::MetaInfo`
Syntax:	`~MetaInfo () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Default destructor .

⌋

8.21.1.3.2 Member Functions

8.21.1.3.2.1 GetContext

[SWS_DM_00979] Definition of API function ara::diag::MetaInfo::GetContext

Upstream requirements: RS_Diag_04170

⌈

Kind:	function	
Header file:	#include "ara/diag/meta_info.h"	
Scope:	`class ara::diag::MetaInfo`	
Syntax:	`Context GetContext () const noexcept;`	
Return value:	Context	Returns the context.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get the context of the invocation.	

⌋

8.21.1.3.2.2 GetValue

[SWS_DM_00978] Definition of API function ara::diag::MetaInfo::GetValue

Upstream requirements: RS_Diag_04170

⌈

Kind:	function	
Header file:	#include "ara/diag/meta_info.h"	
Scope:	`class ara::diag::MetaInfo`	
Syntax:	`ara::core::Optional< ara::core::StringView > GetValue (ara::core::StringView key) const noexcept;`	
Parameters (in):	key	identification of value to be returned
Return value:	ara::core::Optional< ara::core::StringView >	Returns value for the given key.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get the metainfo value for a given key.	

⌋

8.22 Header: ara/diag/release_handler.h

8.22.1 Class: ReleaseHandler

[SWS_DM_01340] Definition of API class ara::diag::ReleaseHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	class
Header file:	#include "ara/diag/release_handler.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	ReleaseHandler
Syntax:	`class ReleaseHandler final {...};`
Description:	ReleaseHandler contains a shared state if the processing should be canceled .

⌋

8.22.1.1 Public Member Types

8.22.1.1.1 Type Alias: ReleaseHandlerSetNotifier

[SWS_DM_02079] Definition of API type ara::diag::ReleaseHandler::ReleaseHandlerSetNotifier

Upstream requirements: RS_Diag_04135

⌈

Kind:	type alias
Header file:	#include "ara/diag/release_handler.h"
Scope:	`class ara::diag::ReleaseHandler`
Symbol:	ReleaseHandlerSetNotifier
Syntax:	`using ReleaseHandlerSetNotifier = std::function<void(void)>;`
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the shared resource can be freed. .

⌋

8.22.1.2 Public Member Functions

8.22.1.2.1 Special Member Functions

8.22.1.2.1.1 Copy Constructor

[SWS_DM_01534] Definition of API function ara::diag::ReleaseHandler::ReleaseHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/release_handler.h"
Scope:	`class ara::diag::ReleaseHandler`
Syntax:	`ReleaseHandler (ReleaseHandler const &)=delete;`
Description:	Copy constructor of ReleaseHandler cannot be used.

⌋

8.22.1.2.1.2 Default Constructor

[SWS_DM_01530] Definition of API function ara::diag::ReleaseHandler::ReleaseHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/release_handler.h"
Scope:	`class ara::diag::ReleaseHandler`
Syntax:	`ReleaseHandler ()=delete;`
Description:	Default constructor of ReleaseHandler cannot be used .

⌋

8.22.1.2.1.3 Move Constructor

[SWS_DM_01532] Definition of API function ara::diag::ReleaseHandler::ReleaseHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/release_handler.h"	
Scope:	`class ara::diag::ReleaseHandler`	
Syntax:	`ReleaseHandler (ReleaseHandler &&other) noexcept;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs instance of class.	

⌋

8.22.1.2.1.4 Move Assignment Operator

[SWS_DM_01533] Definition of API function ara::diag::ReleaseHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/release_handler.h"	
Scope:	`class ara::diag::ReleaseHandler`	
Syntax:	`ReleaseHandler & operator= (ReleaseHandler &&other) & noexcept;`	
Parameters (out):	other	The other object
Return value:	ReleaseHandler &	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns instance of class.	

⌋

8.22.1.2.1.5 Copy Assignment Operator

[SWS_DM_01535] Definition of API function ara::diag::ReleaseHandler::operator=

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/release_handler.h"
Scope:	`class ara::diag::ReleaseHandler`
Syntax:	`ReleaseHandler & operator= (ReleaseHandler const &)=delete;`
Description:	Copy assignment operator of ReleaseHandler cannot be used.

⌋

8.22.1.2.1.6 Destructor

[SWS_DM_01531] Definition of API function ara::diag::ReleaseHandler::~ReleaseHandler

Upstream requirements: RS_Diag_04135

⌈

Kind:	function
Header file:	#include "ara/diag/release_handler.h"
Scope:	`class ara::diag::ReleaseHandler`
Syntax:	`~ReleaseHandler () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Default destructor.

⌋

8.22.1.2.2 Member Functions

8.22.1.2.2.1 MayRelease

[SWS_DM_01536] Definition of API function ara::diag::ReleaseHandler::MayRelease

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/release_handler.h"	
Scope:	`class ara::diag::ReleaseHandler`	
Syntax:	`bool MayRelease () const noexcept;`	
Return value:	bool	True in if the shared resource is no longer in use, False otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reports whether the shared resource is no longer in use.	

⌋

8.22.1.2.2.2 SetNotifier

[SWS_DM_01537] Definition of API function ara::diag::ReleaseHandler::SetNotifier

Upstream requirements: RS_Diag_04135

⌈

Kind:	function	
Header file:	#include "ara/diag/release_handler.h"	
Scope:	`class ara::diag::ReleaseHandler`	
Syntax:	`void SetNotifier (ReleaseHandlerSetNotifier notifier) noexcept;`	
Parameters (in):	notifier	Notification function that is called upon releasing the shared resource
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Registering a notifier function which is called if the shared resource can be freed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.23 Header: ara/diag/security_access.h

8.23.1 Non-Member Types

8.23.1.1 Enumeration: KeyCompareResultType

[SWS_DM_00760] Definition of API enum ara::diag::KeyCompareResultType

Upstream requirements: RS_Diag_04005

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/security_access.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	KeyCompareResultType	
Underlying type:	std::uint8_t	
Syntax:	`enum class KeyCompareResultType : std::uint8_t {...};`	
Values:	kKeyValid= 0x00	Key is valid.
	kKeyInvalid= 0x01	Key is invalid.
Description:	Represents the status of the key compare.	

⌋

8.23.2 Class: SecurityAccess

[SWS_DM_00761] Definition of API class ara::diag::SecurityAccess

Upstream requirements: RS_Diag_04005

⌈

Kind:	class
Header file:	#include "ara/diag/security_access.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SecurityAccess
Syntax:	`class SecurityAccess {...};`
Description:	DiagnosticSecurityAccessInterface.

⌋

8.23.2.1 Public Member Functions

8.23.2.1.1 Special Member Functions

8.23.2.1.1.1 Move Constructor

[SWS_DM_01618] Definition of API function ara::diag::SecurityAccess::SecurityAccess

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`SecurityAccess (SecurityAccess &&) noexcept=delete;`
Description:	Move constructor of SecurityAccess.

⌋

8.23.2.1.1.2 Move Assignment Operator

[SWS_DM_01616] Definition of API function ara::diag::SecurityAccess::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`SecurityAccess & operator= (SecurityAccess &&)=delete;`
Description:	Move assignment operator of SecurityAccess.

⌋

8.23.2.1.1.3 Destructor

[SWS_DM_00763] Definition of API function ara::diag::SecurityAccess::~SecurityAccess

Upstream requirements: RS_AP_00134, RS_Diag_04005

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`virtual ~SecurityAccess () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SecurityAccess .

⌋

8.23.2.1.2 Constructors

8.23.2.1.2.1 SecurityAccess

[SWS_DM_00762] Definition of API function ara::diag::SecurityAccess::SecurityAccess

Upstream requirements: RS_AP_00137, RS_Diag_04005

⌈

Kind:	function	
Header file:	#include "ara/diag/security_access.h"	
Scope:	`class ara::diag::SecurityAccess`	
Syntax:	`explicit SecurityAccess (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticSecurity AccessInterface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticSecurityLevelPortMapping` needs to be typed by a `DiagnosticSecurityLevelInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of SecurityAccess.	

⌋

8.23.2.1.2.2 SecurityAccess

[SWS_DM_01617] Definition of API function ara::diag::SecurityAccess::SecurityAccess

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`SecurityAccess (SecurityAccess &)=delete;`
Description:	SecurityAccess shall be a single not copy-able instance.

⌋

8.23.2.1.3 Member Functions

8.23.2.1.3.1 CompareKey

[SWS_DM_00765] Definition of API function ara::diag::SecurityAccess::CompareKey

Upstream requirements: RS_AP_00138, RS_Diag_04005, RS_Diag_04170, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/security_access.h"	
Scope:	`class ara::diag::SecurityAccess`	
Syntax:	`virtual ara::core::Future< KeyCompareResultType > CompareKey (ara::core::Span< const std::uint8_t > key, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	key	The key to be validated, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< Key CompareResultType >	Result of the key validation.
Exception Safety:	exception safe	

▽

△

Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	This method is called, when a diagnostic request has been finished, to notify about the outcome.	

⌋

8.23.2.1.3.2 GetSeed

[SWS_DM_00764] Definition of API function ara::diag::SecurityAccess::GetSeed

Status: DRAFT
Upstream requirements: RS_AP_00138, RS_Diag_04005, RS_Diag_04170, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/security_access.h"	
Scope:	`class ara::diag::SecurityAccess`	
Syntax:	`virtual ara::core::Future< ara::core::Vector< std::uint8_t > > GetSeed (ara::core::Span< const std::uint8_t > securityAccessDataRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	securityAccessData Record	Security Access payload, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< ara::core::Vector< std::uint8_t > >	provided seed
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for any request messsage.	

⌋

8.23.2.1.3.3 Offer

[SWS_DM_00766] Definition of API function ara::diag::SecurityAccess::Offer

Upstream requirements: RS_AP_00139, RS_Diag_04005, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/security_access.h"	
Scope:	`class ara::diag::SecurityAccess`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.23.2.1.3.4 StopOffer

[SWS_DM_00767] Definition of API function ara::diag::SecurityAccess::StopOffer

Upstream requirements: RS_Diag_04005

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.23.2.1.3.5 operator=

[SWS_DM_01615] Definition of API function ara::diag::SecurityAccess::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/security_access.h"
Scope:	`class ara::diag::SecurityAccess`
Syntax:	`SecurityAccess & operator= (SecurityAccess &)=delete;`
Description:	SecurityAccess shall be a single not assignable instance.

⌋

8.24 Header: ara/diag/service_validation.h

8.24.1 Non-Member Types

8.24.1.1 Enumeration: ConfirmationStatusType

[SWS_DM_00770] Definition of API enum ara::diag::ConfirmationStatusType

Upstream requirements: RS_Diag_04199

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/service_validation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	ConfirmationStatusType	
Underlying type:	std::uint8_t	
Syntax:	`enum class ConfirmationStatusType : std::uint8_t {...};`	
Values:	kResPosOk= 0x00	Positive response has been sent out successfully.
	kResPosNotOk= 0x01	Positive response has not been sent out successfully.
	kResNegOk= 0x02	Negative response has been sent out successfull.
	kResNegNotOk= 0x03	Negative response has not been sent out successfully.
	kResPosSuppressed= 0x04	Positive answer suppressed.
	kResNegSuppressed= 0x05	Negative answer suppressed.
	kCanceled= 0x06	Processing is canceled.

▽

△

	kNoProcessingNo Response= 0x07	Processing rejected in Validation.
Description:	Represents the status of the service processing.	

⌋

8.24.2 Class: ServiceValidation

[SWS_DM_00771] Definition of API class ara::diag::ServiceValidation

Upstream requirements: RS_Diag_04199

⌈

Kind:	class
Header file:	#include "ara/diag/service_validation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	ServiceValidation
Syntax:	`class ServiceValidation {...};`
Description:	DiagnosticServiceValidationInterface.

⌋

8.24.2.1 Public Member Functions

8.24.2.1.1 Special Member Functions

8.24.2.1.1.1 Move Constructor

[SWS_DM_01690] Definition of API function ara::diag::ServiceValidation::ServiceValidation

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`ServiceValidation (ServiceValidation &&)=delete;`
Description:	Move constructor of ServiceValidation.

⌋

8.24.2.1.1.2 Move Assignment Operator

[SWS_DM_01688] Definition of API function ara::diag::ServiceValidation::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`ServiceValidation & operator= (ServiceValidation &&)=delete;`
Description:	Move assignment operator of ServiceValidation.

⌋

8.24.2.1.1.3 Destructor

[SWS_DM_00773] Definition of API function ara::diag::ServiceValidation::~ServiceValidation

Upstream requirements: RS_AP_00134, RS_Diag_04199

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`virtual ~ServiceValidation () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ServiceValidation .

⌋

8.24.2.1.2 Constructors

8.24.2.1.2.1 ServiceValidation

[SWS_DM_00772] Definition of API function ara::diag::ServiceValidation::ServiceValidation

Upstream requirements: RS_AP_00137, RS_Diag_04199

⌈

Kind:	function	
Header file:	#include "ara/diag/service_validation.h"	
Scope:	`class ara::diag::ServiceValidation`	
Syntax:	`explicit ServiceValidation (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an `DiagnosticServiceValidationInterface`
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticServiceValidationMapping` needs to be typed by a `DiagnosticServiceValidationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of ServiceValidation. Depending on value of attribute `DiagnosticServiceValidationMapping.category`, the `Validate` is either called in the context of the NRC sequence in chapter 8.7 response implementation rules from `UDS` ISO 14229-1 as a Manufacturer or a Supplier Check.	

⌋

8.24.2.1.2.2 ServiceValidation

[SWS_DM_01689] Definition of API function ara::diag::ServiceValidation::ServiceValidation

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`ServiceValidation (ServiceValidation &)=delete;`
Description:	ServiceValidation shall be a single not copy-able instance.

⌋

8.24.2.1.3 Member Functions

8.24.2.1.3.1 Confirmation

[SWS_DM_00775] Definition of API function ara::diag::ServiceValidation::Confirmation

Upstream requirements: RS_AP_00138, RS_Diag_04170, RS_Diag_04199

⌈

Kind:	function	
Header file:	#include "ara/diag/service_validation.h"	
Scope:	`class ara::diag::ServiceValidation`	
Syntax:	`void Confirmation (ConfirmationStatusType status, const MetaInfo &meta Info) noexcept;`	
Parameters (in):	status	status/outcome of the service processing.
	metaInfo	MetaInfo of the request.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This method is called, when a diagnostic request has been finished, to notify about the outcome.	

⌋

8.24.2.1.3.2 Offer

[SWS_DM_00776] Definition of API function ara::diag::ServiceValidation::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04199

⌈

Kind:	function	
Header file:	#include "ara/diag/service_validation.h"	
Scope:	`class ara::diag::ServiceValidation`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	Returns nothing or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.24.2.1.3.3 StopOffer

[SWS_DM_00777] Definition of API function ara::diag::ServiceValidation::StopOffer

Upstream requirements: RS_Diag_04199

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.24.2.1.3.4 Validate

[SWS_DM_00774] Definition of API function ara::diag::ServiceValidation::Validate

Upstream requirements: RS_AP_00138, RS_Diag_04170, RS_Diag_04199, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/service_validation.h"	
Scope:	`class ara::diag::ServiceValidation`	
Syntax:	`virtual ara::core::Future< void > Validate (ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo) noexcept;`	
Parameters (in):	requestData	Diagnostic request data (including SID), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	MetaInfo of the request.
Return value:	ara::core::Future< void >	Returns nothing or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for any request messsage.	

⌋

8.24.2.1.3.5 operator=

[SWS_DM_01687] Definition of API function ara::diag::ServiceValidation::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/service_validation.h"
Scope:	`class ara::diag::ServiceValidation`
Syntax:	`ServiceValidation & operator= (ServiceValidation &)=delete;`
Description:	ServiceValidation shall be a single not assignable instance.

⌋

8.25 Header: ara/diag/transmit_certificate.h

8.25.1 Class: TransmitCertificate

[SWS_DM_01961] Definition of API class ara::diag::TransmitCertificate

Upstream requirements: RS_Diag_04251

⌈

Kind:	class
Header file:	#include "ara/diag/transmit_certificate.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	TransmitCertificate
Syntax:	`class TransmitCertificate {...};`
Description:	Class to implement the subfunction TransmitCertificate of UDS service Authentictaion as interface to application. This class implements a so called generic interface, where the application has means to handle one ore more certificateEvaluationId transfered certificates.

⌋

8.25.1.1 Public Member Functions

8.25.1.1.1 Special Member Functions

8.25.1.1.1.1 Move Constructor

[SWS_DM_01963] Definition of API function ara::diag::TransmitCertificate::TransmitCertificate

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`TransmitCertificate (TransmitCertificate &&) noexcept=delete;`
Description:	Move constructor of TransmitCertificate.

⌋

8.25.1.1.1.2 Move Assignment Operator

[SWS_DM_01965] Definition of API function ara::diag::TransmitCertificate::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`TransmitCertificate & operator= (TransmitCertificate &&)=delete;`
Description:	Move assignment operator of TransmitCertificate.

⌋

8.25.1.1.1.3 Destructor

[SWS_DM_01967] Definition of API function ara::diag::TransmitCertificate::~TransmitCertificate

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`virtual ~TransmitCertificate () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of TransmitCertificate .

⌋

8.25.1.1.2 Constructors

8.25.1.1.2.1 TransmitCertificate

[SWS_DM_01964] Definition of API function ara::diag::TransmitCertificate::TransmitCertificate

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`TransmitCertificate (TransmitCertificate &)=delete;`
Description:	Authentication shall be a single not copy-able instance.

⌋

8.25.1.1.2.2 TransmitCertificate

[SWS_DM_01962] Definition of API function ara::diag::TransmitCertificate::TransmitCertificate

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/transmit_certificate.h"	
Scope:	`class ara::diag::TransmitCertificate`	
Syntax:	`explicit TransmitCertificate (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticTransmit Certificate service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticAuthenticationPortMapping` needs to be typed by a `DiagnosticTransmitCertificateInterface`.

▽

△

	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of TransmitCertificate.	

⌋

8.25.1.1.3 Member Functions

8.25.1.1.3.1 Offer

[SWS_DM_01969] Definition of API function ara::diag::TransmitCertificate::Offer

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/transmit_certificate.h"	
Scope:	`class ara::diag::TransmitCertificate`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.25.1.1.3.2 Process

[SWS_DM_01968] Definition of API function ara::diag::TransmitCertificate::Process

Upstream requirements: RS_Diag_04251

⌈

Kind:	function	
Header file:	#include "ara/diag/transmit_certificate.h"	
Scope:	`class ara::diag::TransmitCertificate`	
Syntax:	`virtual ara::core::Future< void > Process (std::uint16_t certificateEvaluationId, ara::core::Span< ara::core::Byte const > certificateData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	certificateEvaluationId	Represents the 16bit certificate evauation ID as part of the UDS request message
	certificateData	Certificate to be verified, transmitted by the tester
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled)..
Return value:	ara::core::Future< void >	void
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidTimePeriod	rollback_semantics
		Date and time of the server does not match the validity period of the Certificate.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidSignature	rollback_semantics
		Signature of the Certificate could not be verified.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidChainOfTrust	rollback_semantics
		Certificate could not be verified against stored information about the issuing authority.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidType	rollback_semantics
		Certificate does not match the current requested use case.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidFormat	rollback_semantics
		Certificate could not be evaluated because the format requirement has not been met.

▽

△

	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidContent	rollback_semantics
		Certificate could not be verified because the content does not match.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidScope	rollback_semantics
		The scope of the Certificate does not match the contents of the server.
	ara::diag::DiagUdsNrcErrc::kCertificateVerificationFailedInvalidCertificateRevoke	rollback_semantics
		Certificate received from client is invalid, because the server has revoked access for some reason.
	ara::diag::DiagUdsNrcErrc::kOwnershipVerificationFailed	rollback_semantics
		Delivered Ownership does not match the provided challenge or could not verified with the own private key.
Description:	This method is used to process certificates send by a diagnostic tester, via the UDS subfucntion TransmitCertificate inside the application. There is no specific semantics for the certificate and it is left to the application to process the received certificate and derive the needed actions.	

⌋

8.25.1.1.3.3 StopOffer

[SWS_DM_01970] Definition of API function ara::diag::TransmitCertificate::StopOffer

Upstream requirements: RS_Diag_04251

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.25.1.1.3.4 operator=

[SWS_DM_01966] Definition of API function ara::diag::TransmitCertificate::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/transmit_certificate.h"
Scope:	`class ara::diag::TransmitCertificate`
Syntax:	`TransmitCertificate & operator= (TransmitCertificate &)=delete;`
Description:	Authentication shall be a single not assignable instance.

⌋

8.26 Header: ara/diag/upload.h

8.26.1 Class: UploadService

[SWS_DM_00794] Definition of API class ara::diag::UploadService

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	class
Header file:	#include "ara/diag/upload.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	UploadService
Syntax:	`class UploadService {...};`
Description:	Upload service interface.

⌋

8.26.1.1 Public Member Functions

8.26.1.1.1 Special Member Functions

8.26.1.1.1.1 Move Constructor

[SWS_DM_01674] Definition of API function ara::diag::UploadService::UploadService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`UploadService (UploadService &&) noexcept=delete;`
Description:	Move constructor of UploadService.

⌋

8.26.1.1.1.2 Move Assignment Operator

[SWS_DM_01672] Definition of API function ara::diag::UploadService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`UploadService & operator= (UploadService &&)=delete;`
Description:	Move assignment operator of UploadService.

⌋

8.26.1.1.1.3 Destructor

[SWS_DM_00798] Definition of API function ara::diag::UploadService::~UploadService

Upstream requirements: RS_AP_00134, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`virtual ~UploadService () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class UploadService .

⌋

8.26.1.1.2 Constructors

8.26.1.1.2.1 UploadService

[SWS_DM_01673] Definition of API function ara::diag::UploadService::UploadService

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`UploadService (UploadService &)=delete;`
Description:	UploadService shall be a single not copy-able instance.

⌋

8.26.1.1.2.2 UploadService

[SWS_DM_00797] Definition of API function ara::diag::UploadService::UploadService

Upstream requirements: RS_AP_00137, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	`class ara::diag::UploadService`	
Syntax:	`explicit UploadService (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticUploadInterface` needs to be referenced by a `DiagnosticServiceGenericMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Class for an UploadService.	

⌋

8.26.1.1.3 Member Functions

8.26.1.1.3.1 Offer

[SWS_DM_00802] Definition of API function ara::diag::UploadService::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	`class ara::diag::UploadService`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.26.1.1.3.2 RequestUpload

[SWS_DM_00799] Definition of API function ara::diag::UploadService::Request Upload

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	`class ara::diag::UploadService`	
Syntax:	`virtual ara::core::Future< void > RequestUpload (std::uint8_t data FormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	dataFormatIdentifier	UDS dataFormat Identifier
	addressAndLength FormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information

▽

△

	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RequestUpload.	

⌋

8.26.1.1.3.3 RequestUploadExit

[SWS_DM_00801] Definition of API function ara::diag::UploadService::RequestUploadExit

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	`class ara::diag::UploadService`	
Syntax:	`virtual ara::core::Future< OperationOutput > RequestUploadExit (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	transferRequest ParameterRecord	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).

▽

△

Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to OperationOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for RequestTransferExit.	

⌋

8.26.1.1.3.4 StopOffer

[SWS_DM_00803] Definition of API function ara::diag::UploadService::StopOffer

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.26.1.1.3.5 UploadData

[SWS_DM_00800] Definition of API function ara::diag::UploadService::UploadData

Upstream requirements: RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196

⌈

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	`class ara::diag::UploadService`	
Syntax:	`virtual ara::core::Future< OperationOutput > UploadData (std::size_t numBytesToReturn, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	numBytesToReturn	number of bytes DM accepts (due to its internal buffer) for this chunk.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to OperationOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in OperationOutput.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagUdsNrcErrc	rollback_semantics
		UDS NRC according to ISO 14229-1
Description:	Called for TransferData following a previous RequestUpload.	

⌋

8.26.1.1.3.6 operator=

[SWS_DM_01671] Definition of API function ara::diag::UploadService::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	`class ara::diag::UploadService`
Syntax:	`UploadService & operator= (UploadService &)=delete;`
Description:	UploadService shall be a single not assignable instance.

⌋

8.26.2 Struct: OperationOutput

[SWS_DM_00795] Definition of API class ara::diag::UploadService::OperationOutput

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	struct
Header file:	#include "ara/diag/upload.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::UploadService`
Symbol:	OperationOutput
Syntax:	`struct OperationOutput {...};`
Description:	Response data of positive respone message.

⌋

8.26.2.1 Public Member Variables

8.26.2.1.1 responseData

[SWS_DM_00796] Definition of API variable ara::diag::UploadService::Operation Output::responseData

Upstream requirements: RS_Diag_04033, RS_Diag_04196

⌈

Kind:	variable
Header file:	#include "ara/diag/upload.h"
Scope:	`struct ara::diag::UploadService::OperationOutput`
Symbol:	responseData
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> responseData;`
Description:	Content of positive respone message (without SID) Depending on the operation (e.g.: UploadData, RequestUploadExit) the expectation, what responseData shall contain (where it starts in the positive response) might differ. See doc of corresponding operation.

⌋

8.27 Header: {<data-element-directory-path>}/{<data-element-shortname-lower>}.h

[SWS_DM_80021] `Diagnostic DataElement Header File`: file name, includes and multiple inclusion guard

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	Header File	
Syntax:	`{<data-element-directory-path>}/{<data-element-shortname-lower>}.h`	
Description:	For each modeled `DiagnosticDataElementInterface` a `Diagnostic DataElement Header File` is generated according to this directory path/file name convention and shall: 1. Insert a multiple inclusion guard around the whole header file as per [SWS_CORE_90002]	
Descriptors:	`{<data-element-directory-path>}`	as per [SWS_DM_80021] whereby: for each inner namespace in the hierarchy, an inner directory shall be created to contain the header file
	`{<data-element-shortname-lower>}`	as per `DiagnosticDataElementInterface.shortName` converted to lower-case.

▽

△

Example:	

	`// File=n/n_plus_1/n_plus_2/si_data-element.h (1)`
	`#ifndef N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#define N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#include ".../path/to/si_common.h" (3)`
	`...`
	`#endif // N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`

⌋

8.27.1 Namespaces

8.27.1.1 {<namespace-list-data-element>}

[SWS_DM_80022] `Diagnostic DataElement Header File:` **service namespace**

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	namespace	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	--	
Syntax:	`namespace {<namespace-list-data-element>}`	
Description:	The generator shall use the `SymbolProps` aggregated in the role `PortInterface`.`namespace`. For each `namespace` in the **ordered** list: `namespace`[N+1] shall be an inner namespace of `namespace`[N] converted to lower-case.	
Descriptors:	`{<namespace-list-data-element>}`	as per `namespace` in the **ordered** list: `namespace`[N+1] shall be an inner namespace of `namespace`[N] converted to lower-case.

⌋

8.27.2 Class: {<data-element-interface-name>}

[SWS_DM_00603] Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name>}

Status: DRAFT
Upstream requirements: RS_Main_01002, RS_Diag_04169

⌈

Kind:	class	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Forwarding header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}_fwd.h"	
Scope:	namespace {<namespace-list-data-element>}	
Symbol:	{<data-element-interface-name>}	
Syntax:	class {<data-element-interface-name>} {...};	
Description:	DiagnosticDataElementInterface class	
Descriptors:	{<data-element-interface-name>}	The DiagnosticDataElementInterface. shortName converted to upper camel-case letters

⌋

8.27.2.1 Public Member Variables

8.27.2.1.1 {<data-element-out-arg-symbol>}

[SWS_DM_80023] Definition of API variable {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-out-arg-symbol>}

Status: DRAFT
Upstream requirements: RS_Diag_04224, RS_Diag_04169

⌈

Kind:	variable
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}
Symbol:	{<data-element-out-arg-symbol>}
Type:	{<data-element-out-arg-type>}
Syntax:	{<data-element-out-arg-type>} {<data-element-out-arg-symbol>};
Description:	Member declaration representing an *out* argument in an Output.

▽

△

Descriptors:	{<data-element-out-arg-type>}	The ClientServerOperation. argument. type, mapped to a C++ data type according to [14].
	{<data-element-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype. shortName

⌋

8.27.2.2 Public Member Functions

8.27.2.2.1 Special Member Functions

8.27.2.2.1.1 Move Assignment Operator

[SWS_DM_01660] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	{<data-element-interface-name>} & operator= ({<data-element-interface-name>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.1.2 Copy Assignment Operator

[SWS_DM_01659] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	{<data-element-interface-name>} & operator= (const {<data-element-interface-name>} &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.1.3 Copy Constructor

[SWS_DM_01661] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	{<data-element-interface-name>} (const {<data-element-interface-name>} &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.1.4 Move Constructor

[SWS_DM_01662] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	{<data-element-interface-name>} ({<data-element-interface-name>} &&other)=delete;	
Description:	Move constructor	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.1.5 Destructor

[SWS_DM_00588] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::~{<data-element-interface-name>}

Status: DRAFT

Upstream requirements: RS_Diag_04169, RS_AP_00134

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	virtual ~{<data-element-interface-name>} () noexcept;	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Destruction of a DataElement	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.2 Constructors

8.27.2.2.2.1 {<data-element-interface-name>}

[SWS_DM_00587] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-interface-name>}

Upstream requirements: RS_AP_00137, RS_AP_00137

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	{<data-element-interface-name>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	specifies if interface is callable fully- or non-reentrant
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Violations:	InstanceSpecifierMappingIntegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticDataElementInterface needs to be referenced by a DiagnosticDataPortMapping.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Construct a DataElement from an ara::core::InstanceSpecifier	
Descriptors:	{<data-element-interface-name>}	As per {<data-element-interface-name>} in [SWS_DM_00603]

⌋

8.27.2.2.3 Member Functions

8.27.2.2.3.1 Offer

[SWS_DM_00597] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	ara::core::Result< void > Offer () noexcept;	
Return value:	ara::core::Result< void >	• If successful: an ara::core::Result containing a ara::core::Result::value_type • If unsuccessful: an ara::core::Result containing an ara::core::Result::error_type i.e. a corresponding ara::diag::DiagOfferErrc
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Errors:	DiagOfferErrc::kAlready Offered	--
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

⌋

8.27.2.2.3.2 Read

[SWS_DM_00596] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::Read

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Syntax:	virtual ara::core::Future< {<data-element-name-upper-camel>}Output > Read (const ara::diag::MetaInfo &metaInfo, ara::diag::Cancellation Handler cancellationHandler)=0 noexcept;	
Parameters (in):	metaInfo	MetaInfo of the request.

▽

△

	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< {<data-element-name-upper-camel>}Output >	• If successful: an `ara::core::Future` containing an `Output` object as per [SWS_DM_00581] • If unsuccessful: an `ara::core::Future` containing a corresponding `ara::diag::DiagUdsNrcErrc` or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a Start DataElement (`DiagnosticDataElementInterface` in the role `start`)	
Descriptors:	{<data-element-name-upper-camel>}	Name of a DataElement created from the `ClientServerOperation`. `shortName` defined in the `DiagnosticDataElementInterface` in the role `start` converted to upper camel-case letters.

Example:	`// Example:`
	`virtual ara::core::Future<SomeMethodOutput>`
	`Read(`
	`  const ara::diag::MetaInfo& metaInfo,`
	`  ara::diag::CancellationHandler cancellationHandler`
	`) = 0;`	

⌋

8.27.2.2.3.3 StopOffer

[SWS_DM_00617] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::StopOffer

Upstream requirements: RS_Diag_04169, RS_Main_01002

⌈

Kind:	function
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"
Scope:	`class {<namespace-list-data-element>}::{<data-element-interface-name>}`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	This StopOffer will disable the forwarding of request messages from DM

⌋

8.27.3 Struct: {<data-element-name-upper-camel>}Output

[SWS_DM_00580] Definition of API class `{<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-name-upper-camel>}`Output

Status: DRAFT
Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	struct	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Forwarding header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}_fwd.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name>}	
Symbol:	{<data-element-name-upper-camel>}Output	
Syntax:	struct {<data-element-name-upper-camel>}Output {...};	
Description:	Structure wrapping the *out* arguments for a Read	
Descriptors:	{<data-element-name-upper-camel>}	As per {<data-element-name-upper-camel>} in [SWS_DM_00596]
	{<data-element-out-args-members-ordered>}	**Shown as "..." in Syntax**. Each element is an argument in the ordered list of arguments, with direction == out **or** direction == inout, where [SWS_DM_80023] applies for each.

⌋

8.28 Header: {<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h

[SWS_DM_80011] `Diagnostic DataIdentifier Header File`: file name, includes and multiple inclusion guard

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	Header File
Syntax:	{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h
Description:	For each modeled DiagnosticDataIdentifierInterface a Diagnostic DataIdentifier Header File is generated according to this directory path/file name convention and shall: 1. Insert a multiple inclusion guard around the whole header file as per [SWS_CORE_90002]

▽

△

Descriptors:	{<data-identifier-directory-path>}	as per [SWS_DM_80011] whereby: for each inner namespace in the hierarchy, an inner directory shall be created to contain the header file
	{<data-identifier-shortname-lower>}	as per DiagnosticDataIdentifierInterface. shortName converted to lower-case.

Example:	`// File=n/n_plus_1/n_plus_2/si_data identifier.h (1)`
	`#ifndef N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#define N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#include ".../path/to/si_common.h" (3)`
	`...`
	`#endif // N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`	

⌋

8.28.1 Namespaces

8.28.1.1 {<namespace-list-data-identifier>}

[SWS_DM_80012] Diagnostic DataIdentifier Header File: service namespace

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	namespace
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"
Scope:	--
Syntax:	namespace {<namespace-list-data-identifier>}
Description:	The generator shall use the SymbolProps aggregated in the role PortInterface. namespace. For each namespace in the **ordered** list: namespace[N+1] shall be an inner namespace of namespace[N] converted to lower-case.
Descriptors:	{<namespace-list-data-identifier>} — as per namespace in the **ordered** list: namespace[N+1] shall be an inner namespace of namespace[N] converted to lower-case.

⌋

8.28.2 Class: {<data-identifier-interface-name>}

[SWS_DM_00601] Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}

Status: DRAFT
Upstream requirements: RS_Main_01002, RS_Diag_04169

⌈

Kind:	class
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"
Forwarding header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}_fwd.h"
Scope:	namespace {<namespace-list-data-identifier>}
Symbol:	{<data-identifier-interface-name>}
Syntax:	class {<data-identifier-interface-name>} {...};
Description:	DiagnosticDataIdentifierInterface class
Descriptors:	{<data-identifier-interface-name>} — The DiagnosticDataIdentifierInterface. shortName converted to upper camel-case letters

⌋

8.28.2.1 Public Member Variables

8.28.2.1.1 {<data-identifier-out-arg-symbol>}

[SWS_DM_80013] Definition of API variable {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-out-arg-symbol>}

Status: DRAFT
Upstream requirements: RS_Diag_04224, RS_Diag_04169

⌈

Kind:	variable
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}
Symbol:	{<data-identifier-out-arg-symbol>}
Type:	{<data-identifier-out-arg-type>}
Syntax:	{<data-identifier-out-arg-type>} {<data-identifier-out-arg-symbol>};
Description:	Member declaration representing an *out* argument in an Output.

▽

△

Descriptors:	{<data-identifier-out-arg-type>}	The ClientServerOperation. argument. type, mapped to a C++ data type according to [14].
	{<data-identifier-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype. shortName

⌋

8.28.2.2 Public Member Functions

8.28.2.2.1 Special Member Functions

8.28.2.2.1.1 Move Assignment Operator

[SWS_DM_01664] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	{<data-identifier-interface-name>} & operator= ({<data-identifier-interface-name>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.1.2 Copy Assignment Operator

[SWS_DM_01663] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	{<data-identifier-interface-name>} & operator= (const {<data-identifier-interface-name>} &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.1.3 Move Constructor

[SWS_DM_01666] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	{<data-identifier-interface-name>} ({<data-identifier-interface-name>} &&other)=delete;	
Description:	Move constructor	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.1.4 Copy Constructor

[SWS_DM_01665] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	{<data-identifier-interface-name>} (const {<data-identifier-interface-name>} &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.1.5 Destructor

[SWS_DM_00586] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::~{<data-identifier-interface-name>}

Status: DRAFT

Upstream requirements: RS_Diag_04169, RS_AP_00134

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	virtual ~{<data-identifier-interface-name>} ();	
Exception Safety:	not exception safe	
Thread Safety:	implementation defined	
Description:	Destruction of a DataIdentifier	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.2 Constructors

8.28.2.2.2.1 {<data-identifier-interface-name>}

[SWS_DM_00585] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}

Upstream requirements: RS_AP_00137, RS_AP_00137

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	{<data-identifier-interface-name>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	specifies if interface is callable fully- or non-reentrant
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Violations:	InstanceSpecifierMappingIntegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticDataIdentifierInterface needs to be referenced by a DiagnosticDataPortMapping.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Construct a DataIdentifier from an ara::core::InstanceSpecifier	
Descriptors:	{<data-identifier-interface-name>}	As per {<data-identifier-interface-name>} in [SWS_DM_00601]

⌋

8.28.2.2.3 Member Functions

8.28.2.2.3.1 Offer

[SWS_DM_00599] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	ara::core::Result< void > Offer () noexcept;	
Return value:	ara::core::Result< void >	• If successful: an ara::core::Result containing a ara::core::Result::value_type • If unsuccessful: an ara::core::Result containing an ara::core::Result::error_type i.e. a corresponding ara::diag::DiagOfferErrc
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Errors:	DiagOfferErrc::kAlready Offered	--
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

⌋

8.28.2.2.3.2 Read

[SWS_DM_00640] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::Read

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	virtual ara::core::Future< {<data-identifier-name-upper-camel>}Output > Read (const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0 noexcept;	
Parameters (in):	metaInfo	MetaInfo of the request.

▽

△

	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< {<data-identifier-name-upper-camel>}Output >	• If successful: an `ara::core::Future` containing an `Output` object as per [SWS_DM_00640] • If unsuccessful: an `ara::core::Future` containing a corresponding `ara::diag::DiagUdsNrcErrc` or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a reading DataIdentifier (`DiagnosticDataIdentifierInterface` in the role `read`)	
Descriptors:	{<data-identifier-name-upper-camel>}	Name of a DataIdentifier created from the `ClientServerOperation`. `shortName` defined in the `DiagnosticDataIdentifierInterface` in the role `start` converted to upper camel-case letters.

Example:	`// Example:`
	`virtual ara::core::Future<SomeMethodOutput>`
	`Read(`
	`  const ara::diag::MetaInfo& metaInfo,`
	`  ara::diag::CancellationHandler cancellationHandler`
	`) = 0;`	

⌋

8.28.2.2.3.3 StopOffer

[SWS_DM_00600] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::StopOffer

Upstream requirements: RS_Diag_04169, RS_Main_01002

⌈

Kind:	function
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	This StopOffer will disable the forwarding of request messages from DM

⌋

8.28.2.2.3.4 Write

[SWS_DM_00598] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::Write

Status: DRAFT
Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Syntax:	virtual ara::core::Future< void > Write ({<data-identifier-in-arg-derived-type-0toN>} {<data-identifier-in-arg-symbol-0toN>}, const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0 noexcept;	
Parameters (in):	{<data-identifier-in-arg-symbol-0toN>}	The symbol name of method argument[0..N] in the ordered list of method arguments, with ClientServerOperation. direction == in **or** direction == inout given by ClientServerOperation. ArgumentDataPrototype[0..N]. shortName
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< void >	• If successful: an ara::core::Future containing an void • If unsuccessful: an ara::core::Future containing a corresponding ara::diag::DiagUdsNrcErrc or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a writing DataIdentifier (DiagnosticDataIdentifierInterface in the role write)	

▽

△

Descriptors:	{<derived-type>}	The C++ data type shall be derived according to the following rules: 1. If the (CppImplementationDataType. category ==VALUE **or** a (CppImplementationDataType. category ==TYPE_REFERENCE which transitively type-resolves to a CppImplementationDataType. category ==VALUE) **and** the CppImplementationDataType. shortName is either: • int8_t: as per [SWS_APT_00001] • int16_t: as per [SWS_APT_00004] • int32_t: as per [SWS_APT_00007] • int64_t: as per [SWS_APT_00010] • uint8_t: as per [SWS_APT_00022] • uint16_t: as per [SWS_APT_00025] • uint32_t: as per [SWS_APT_00028] • uint64_t: as per [SWS_APT_00031] • bool: as per [SWS_APT_00049] • float: as per [SWS_APT_00043] • double: as per [SWS_APT_00046] in [15], then {<derived-type>} shall be unchanged, i.e. shall be the mapped C++ data type according to [14]. 2. Otherwise {<derived-type>} shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [16]), to the mapped C++ data type according to [14].
	{<data-identifier-in-arg-derived-type-0toN>}	The data type of method argument[0..N] in the ordered list of method arguments, with ClientServerOperation. direction == in **or** direction == inout given by ClientServerOperation. argument[0..N]. type, derived as per {<derived-type>}
Example:		

```
// Example:
virtual ara::core::Future<void>
Write(
  // {<derived-type>} sub-clause 1:
  std::int8_t inArg0,

  // {<derived-type>} sub-clause 1:
  //     e.g. if inArg1 would type-resolve to std::uint64_t
  ns1::ns2::sometype_t inArg1,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArg2 would type-resolve to ara::core::map
  const ns1::ns2::anothertype_t& inArg2,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArgN would type-resolve to struct
  const ns1::ns2::furthertype_t& inArgN,
  const ara::diag::MetaInfo& metaInfo,
  ara::diag::CancellationHandler cancellationHandler
) = 0;
```

⌋

8.28.3 Struct: {<data-identifier-name-upper-camel>}Read

[SWS_DM_00579] Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-name-upper-camel>}Read

Status: DRAFT
Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	struct	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Forwarding header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}_fwd.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}	
Symbol:	{<data-identifier-name-upper-camel>}Read	
Syntax:	struct {<data-identifier-name-upper-camel>}Read {...};	
Description:	Structure wrapping the *out* arguments for a Read	
Descriptors:	{<data-identifier-name-upper-camel>}	As per {<data-identifier-name-upper-camel>} in [SWS_DM_00640]
	{<data-identifier-out-args-members-ordered>}	**Shown as "..." in Syntax**. Each element is an argument in the ordered list of arguments, with direction == out **or** direction == inout, where [SWS_DM_80013] applies for each.

⌋

8.29 Header: {<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h

[SWS_DM_80001] Diagnostic Routine Header File: file name, includes and multiple inclusion guard

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	Header File
Syntax:	{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h
Description:	For each modeled DiagnosticRoutineInterface a Diagnostic Routine Header File is generated according to this directory path/file name convention and shall: 1. Insert a multiple inclusion guard around the whole header file as per [SWS_CORE_90002]

▽

△

Descriptors:	{<routine-interface-directory-path>}	as per [SWS_DM_80001] whereby: for each inner namespace in the hierarchy, an inner directory shall be created to contain the header file
	{<routine-interface-shortname-lower>}	as per DiagnosticRoutineInterface. shortName converted to lower-case.

Example:	`// File=n/n_plus_1/n_plus_2/si_skeleton.h (1)`
	`#ifndef N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#define N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`
	`#include ".../path/to/si_common.h" (3)`
	`...`
	`#endif // N_NPLUS1_NPLUS2_SI_ROUTINE_H_ (2)`	

⌋

8.29.1 Namespaces

8.29.1.1 {<routine-interface-hierarchical-namespace-list>}

[SWS_DM_80002] `Diagnostic Routine Header File`: **service namespace**

Status: DRAFT
Upstream requirements: RS_AP_00114

⌈

Kind:	namespace	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	--	
Syntax:	namespace {<routine-interface-hierarchical-namespace-list>}	
Description:	The generator shall use the SymbolProps aggregated in the role PortInterface. namespace. For each namespace in the **ordered** list: namespace[N+1] shall be an inner namespace of namespace[N] converted to lower-case.	
Descriptors:	{<routine-interface-hierarchical-namespace-list>}	as per namespace in the **ordered** list: namespace[N+1] shall be an inner namespace of namespace[N] converted to lower-case.

⌋

8.29.2 Class: {<routine-interface-name>}

[SWS_DM_00604] Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}

Status: DRAFT
Upstream requirements: RS_Main_01002, RS_Diag_04169

⌈

Kind:	class	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Forwarding header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	namespace {<routine-interface-hierarchical-namespace-list>}	
Symbol:	{<routine-interface-name>}	
Syntax:	class {<routine-interface-name>} {...};	
Description:	DiagnosticRoutineInterface class	
Descriptors:	{<routine-interface-name>}	The DiagnosticRoutineInterface. shortName converted to upper camel-case letters

⌋

8.29.2.1 Public Member Variables

8.29.2.1.1 {<routine-interface-request-result-out-arg-symbol>}

[SWS_DM_80005] Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-request-result-out-arg-symbol>}

Status: DRAFT
Upstream requirements: RS_Diag_04224, RS_Diag_04169

⌈

Kind:	variable
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}
Symbol:	{<routine-interface-request-result-out-arg-symbol>}
Type:	{<routine-interface-request-result-out-arg-type>}
Syntax:	{<routine-interface-request-result-out-arg-type>} {<routine-interface-request-result-out-arg-symbol>};
Description:	Member declaration representing an *out* argument in an RequestResultOutput.

▽

△

Descriptors:	{<routine-interface-request-result-out-arg-type>}	The ClientServerOperation. argument. type, mapped to a C++ data type according to [14].
	{<routine-interface-request-result-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype. shortName

⌋

8.29.2.1.2 {<routine-interface-start-out-arg-symbol>}

[SWS_DM_80003] Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-start-out-arg-symbol>}

Status: DRAFT
Upstream requirements: RS_Diag_04224, RS_Diag_04169

⌈

Kind:	variable	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Symbol:	{<routine-interface-start-out-arg-symbol>}	
Type:	{<routine-interface-start-out-arg-type>}	
Syntax:	{<routine-interface-start-out-arg-type>} {<routine-interface-start-out-arg-symbol>};	
Description:	Member declaration representing an *out* argument in an StartOutput.	
Descriptors:	{<routine-interface-start-out-arg-type>}	The ClientServerOperation. argument. type, mapped to a C++ data type according to [14].
	{<routine-interface-start-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype. shortName

⌋

8.29.2.1.3 {<routine-interface-stop-out-arg-symbol>}

[SWS_DM_80004] Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-stop-out-arg-symbol>}

Status: DRAFT
Upstream requirements: RS_Diag_04224, RS_Diag_04169

⌈

Kind:	variable	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Symbol:	{<routine-interface-stop-out-arg-symbol>}	
Type:	{<routine-interface-stop-out-arg-type>}	
Syntax:	{<routine-interface-stop-out-arg-type>} {<routine-interface-stop-out-arg-symbol>};	
Description:	Member declaration representing an *out* argument in an StopOutput.	
Descriptors:	{<routine-interface-stop-out-arg-type>}	The ClientServerOperation. argument. type, mapped to a C++ data type according to [14].
	{<routine-interface-stop-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype. shortName

⌋

8.29.2.2 Public Member Functions

8.29.2.2.1 Special Member Functions

8.29.2.2.1.1 Move Assignment Operator

[SWS_DM_01668] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	{<routine-interface-name>} & operator= ({<routine-interface-name>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<routine-interface-name>}	As per {<routine-interface-name>} in [SWS_DM_00604]

⌋

8.29.2.2.1.2 Copy Assignment Operator

[SWS_DM_01667] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::operator=

Status: DRAFT
Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}
Syntax:	{<routine-interface-name>} & operator= (const {<routine-interface-name>} &other)=delete;
Description:	Copy assignment constructor deletion

▽

△

Descriptors:	{<routine-interface-name>}	As per {<routine-interface-name>} in [SWS_DM_00604]

⌋

8.29.2.2.1.3 Move Constructor

[SWS_DM_01670] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	{<routine-interface-name>} ({<routine-interface-name>} &&other)=delete;	
Description:	Move constructor	
Descriptors:	{<routine-interface-name>}	As per {<routine-interface-name>} in [SWS_DM_00604]

⌋

8.29.2.2.1.4 Copy Constructor

[SWS_DM_01669] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}

▽

△

Syntax:	`{<routine-interface-name>} (const {<routine-interface-name>} &other)=delete;`	
Description:	Copy constructor deletion	
Descriptors:	`{<routine-interface-name>}`	As per `{<routine-interface-name>}` in [SWS_DM_00604]

⌋

8.29.2.2.1.5 Destructor

[SWS_DM_00590] Definition of API function `{<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::~{<routine-interface-name>}`

Status: DRAFT
Upstream requirements: RS_Diag_04169, RS_AP_00134

⌈

Kind:	function	
Header file:	#include "`{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}`.h"	
Scope:	`class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}`	
Syntax:	`virtual ~{<routine-interface-name>} () noexcept;`	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Destruction of a `DiagnosticRoutine`	
Descriptors:	`{<routine-interface-name>}`	As per `{<routine-interface-name>}` in [SWS_DM_00604]

⌋

8.29.2.2.2 Constructors

8.29.2.2.2.1 {<routine-interface-name>}

[SWS_DM_00589] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}

Upstream requirements: RS_AP_00137, RS_AP_00137

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	{<routine-interface-name>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	specifies if interface is callable fully- or non-reentrant
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Violations:	InstanceSpecifierMappingIntegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticRoutineInterface needs to be referenced by a DiagnosticServiceGenericMapping.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Construct a DiagnosticRoutine from an ara::core::InstanceSpecifier	
Descriptors:	{<routine-interface-name>}	As per {<routine-interface-name>} in [SWS_DM_00604]

⌋

8.29.2.2.3 Member Functions

8.29.2.2.3.1 Offer

[SWS_DM_00594] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169, RS_Diag_04224

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	ara::core::Result< void > Offer () noexcept;	
Return value:	ara::core::Result< void >	• If successful: an ara::core::Result containing a ara::core::Result::value_type • If unsuccessful: an ara::core::Result containing an ara::core::Result::error_type i.e. a corresponding ara::diag::DiagOfferErrc
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Errors:	DiagOfferErrc::kAlready Offered	--
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

⌋

8.29.2.2.3.2 RequestResult

[SWS_DM_00593] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::RequestResult

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}

▽

△

Syntax:	`virtual ara::core::Future< {<routine-interface-name-upper-camel>}RequestResultOutput > RequestResult ({<routine-interface-request-result-in-arg-derived-type-0toN>} {<routine-interface-request-result-in-arg-symbol-0toN>}, const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0 noexcept;`	
Parameters (in):	`{<routine-interface-request-result-in-arg-symbol-0toN>}`	The symbol name of method `argument`[0..N] in the ordered list of `method` arguments, with `ClientServerOperation.direction == in` **or** `direction == inout` given by `ClientServerOperation.ArgumentDataPrototype`[0..N]. `shortName`
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< `{<routine-interface-name-upper-camel>}`RequestResultOutput >	• If successful: an `ara::core::Future` containing an `StartOutput` object as per [SWS_DM_00583] • If unsuccessful: an `ara::core::Future` containing a corresponding `ara::diag::DiagUdsNrcErrc` or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a RequestResult RoutineControl (`DiagnosticRoutineInterface` in the role `RequestResult`)	
Descriptors:	`{<routine-interface-name-upper-camel>}`	Name of a DiagnosticRoutine created from the `ClientServerOperation.shortName` defined in the `DiagnosticRoutineInterface` in the role `start` converted to upper camel-case letters.
	`{<derived-type>}`	The C++ data type shall be derived according to the following rules: 1. If the (`CppImplementationDataType.category ==VALUE` **or** a (`CppImplementationDataType.category ==TYPE_REFERENCE` which transitively type-resolves to a `CppImplementationDataType.category ==VALUE`) **and** the `CppImplementationDataType.shortName` is either: • `int8_t`: as per [SWS_APT_00001] • `int16_t`: as per [SWS_APT_00004] • `int32_t`: as per [SWS_APT_00007] • `int64_t`: as per [SWS_APT_00010] • `uint8_t`: as per [SWS_APT_00022] • `uint16_t`: as per [SWS_APT_00025] • `uint32_t`: as per [SWS_APT_00028] • `uint64_t`: as per [SWS_APT_00031] • `bool`: as per [SWS_APT_00049] • `float`: as per [SWS_APT_00043] • `double`: as per [SWS_APT_00046] in [15], then `{<derived-type>}` shall be unchanged, i.e. shall be the mapped C++ data type according to [14]. 2. Otherwise `{<derived-type>}` shall be a const-qualified lvalue reference (see `[basic.type.qualifier]`, `[basic.lval]` in [16]), to the mapped C++ data type according to [14].

▽

△

	{<routine-interface-request-result-in-arg-derived-type-0toN>}	The data type of method argument[0..N] in the ordered list of method arguments, with ClientServerOperation. direction == in **or** direction == inout given by ClientServerOperation. argument[0..N]. type, derived as per {<derived-type>}
Example:		

```
// Example:
virtual ara::core::Future<SomeMethodOutput>
Start(
  // {<derived-type>} sub-clause 1:
  std::int8_t inArg0,

  // {<derived-type>} sub-clause 1:
  //     e.g. if inArg1 would type-resolve to std::uint64_t
  ns1::ns2::sometype_t inArg1,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArg2 would type-resolve to ara::core::map
  const ns1::ns2::anothertype_t& inArg2,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArgN would type-resolve to struct
  const ns1::ns2::furthertype_t& inArgN,
  const ara::diag::MetaInfo& metaInfo,
  ara::diag::CancellationHandler cancellationHandler
) = 0;
```

⌋

8.29.2.2.3.3 Start

[SWS_DM_00591] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Start

Status: DRAFT
Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	virtual ara::core::Future< {<routine-interface-name-upper-camel>}StartOutput > Start ({<routine-interface-start-in-arg-derived-type-0toN>} {<routine-interface-start-in-arg-symbol-0toN>}, const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0 noexcept;	
Parameters (in):	{<routine-interface-start-in-arg-symbol-0toN>}	The symbol name of method argument[0..N] in the ordered list of method arguments, with ClientServerOperation. direction == in **or** direction == inout given by ClientServerOperation. ArgumentDataPrototype[0..N]. shortName

▽

△

	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< {<routine-interface-name-upper-camel>}Start Output >	• If successful: an `ara::core::Future` containing an `StartOutput` object as per [SWS_DM_00581] • If unsuccessful: an `ara::core::Future` containing a corresponding `ara::diag::DiagUdsNrcErrc` or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a Start RoutineControl (`DiagnosticRoutineInterface` in the role `start`)	
Descriptors:	{<routine-interface-name-upper-camel>}	Name of a DiagnosticRoutine created from the `ClientServerOperation`. `shortName` defined in the `DiagnosticRoutineInterface` in the role `start` converted to upper camel-case letters.
	{<derived-type>}	The C++ data type shall be derived according to the following rules: 1. If the (`CppImplementationDataType`. `category` ==VALUE **or** a (`CppImplementationDataType`. `category` ==TYPE_REFERENCE which transitively type-resolves to a `CppImplementationDataType`. `category` ==VALUE) **and** the `CppImplementationDataType`. `shortName` is either: • `int8_t`: as per [SWS_APT_00001] • `int16_t`: as per [SWS_APT_00004] • `int32_t`: as per [SWS_APT_00007] • `int64_t`: as per [SWS_APT_00010] • `uint8_t`: as per [SWS_APT_00022] • `uint16_t`: as per [SWS_APT_00025] • `uint32_t`: as per [SWS_APT_00028] • `uint64_t`: as per [SWS_APT_00031] • `bool`: as per [SWS_APT_00049] • `float`: as per [SWS_APT_00043] • `double`: as per [SWS_APT_00046] in [15], then {<derived-type>} shall be unchanged, i.e. shall be the mapped C++ data type according to [14]. 2. Otherwise {<derived-type>} shall be a const-qualified lvalue reference (see `[basic.type.qualifier]`, `[basic.lval]` in [16]), to the mapped C++ data type according to [14].
	{<routine-interface-start-in-arg-derived-type-0toN>}	The data type of method `argument`[0..N] in the ordered list of `method` arguments, with `ClientServerOperation`. `direction` == `in` **or** `direction` == `inout` given by `ClientServerOperation`. `argument`[0..N]. `type`, derived as per {<derived-type>}

▽

△

Example:	

```
// Example:
virtual ara::core::Future<SomeMethodOutput>
Start(
  // {<derived-type>} sub-clause 1:
  std::int8_t inArg0,

  // {<derived-type>} sub-clause 1:
  //     e.g. if inArg1 would type-resolve to std::uint64_t
  ns1::ns2::sometype_t inArg1,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArg2 would type-resolve to ara::core::map
  const ns1::ns2::anothertype_t& inArg2,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArgN would type-resolve to struct
  const ns1::ns2::furthertype_t& inArgN,
  const ara::diag::MetaInfo& metaInfo,
  ara::diag::CancellationHandler cancellationHandler
) = 0;
```

⌋

8.29.2.2.3.4 Stop

[SWS_DM_00592] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Stop

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Syntax:	virtual ara::core::Future< {<routine-interface-name-upper-camel>}StopOutput > Stop ({<routine-interface-stop-in-arg-derived-type-0toN>} {<routine-interface-stop-in-arg-symbol-0toN>}, const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0 noexcept;	
Parameters (in):	{<routine-interface-stop-in-arg-symbol-0toN>}	The symbol name of method argument[0..N] in the ordered list of method arguments, with ClientServerOperation. direction == in **or** direction == inout given by ClientServerOperation. ArgumentDataPrototype[0..N]. shortName
	metaInfo	MetaInfo of the request.

▽

△

	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.Set Notifier() (the registered notifier is called if the current conversation is cancelled).
Return value:	ara::core::Future< {<routine-interface-name-upper-camel>}Stop Output >	• If successful: an `ara::core::Future` containing an `StartOutput` object as per [SWS_DM_00582] • If unsuccessful: an `ara::core::Future` containing a corresponding `ara::diag::DiagUdsNrcErrc` or Ap ApplicationError.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Provision of a Stop RoutineControl (`DiagnosticRoutineInterface` in the role `stop`)	
Descriptors:	{<routine-interface-name-upper-camel>}	Name of a DiagnosticRoutine created from the `ClientServerOperation`. `shortName` defined in the `DiagnosticRoutineInterface` in the role `start` converted to upper camel-case letters.
	{<derived-type>}	The C++ data type shall be derived according to the following rules: 1. If the (`CppImplementationDataType`. `category` ==VALUE **or** a (`CppImplementationDataType`. `category` ==TYPE_REFERENCE which transitively type-resolves to a `CppImplementationDataType`. `category` ==VALUE) **and** the `CppImplementationDataType`. `shortName` is either: • `int8_t`: as per [SWS_APT_00001] • `int16_t`: as per [SWS_APT_00004] • `int32_t`: as per [SWS_APT_00007] • `int64_t`: as per [SWS_APT_00010] • `uint8_t`: as per [SWS_APT_00022] • `uint16_t`: as per [SWS_APT_00025] • `uint32_t`: as per [SWS_APT_00028] • `uint64_t`: as per [SWS_APT_00031] • `bool`: as per [SWS_APT_00049] • `float`: as per [SWS_APT_00043] • `double`: as per [SWS_APT_00046] in [15], then {<derived-type>} shall be unchanged, i.e. shall be the mapped C++ data type according to [14]. 2. Otherwise {<derived-type>} shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [16]), to the mapped C++ data type according to [14].
	{<routine-interface-stop-in-arg-derived-type-0toN>}	The data type of method `argument`[0..N] in the ordered list of `method` arguments, with `ClientServerOperation`. `direction` == `in` **or** `direction` == `inout` given by `ClientServerOperation`. `argument`[0..N]. `type`, derived as per {<derived-type>}

▽

△

Example:	

```
// Example:
virtual ara::core::Future<SomeMethodOutput>
Start(
  // {<derived-type>} sub-clause 1:
  std::int8_t inArg0,

  // {<derived-type>} sub-clause 1:
  //     e.g. if inArg1 would type-resolve to std::uint64_t
  ns1::ns2::sometype_t inArg1,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArg2 would type-resolve to ara::core::map
  const ns1::ns2::anothertype_t& inArg2,

  // {<derived-type>} sub-clause 2:
  //     e.g. if inArgN would type-resolve to struct
  const ns1::ns2::furthertype_t& inArgN,
  const ara::diag::MetaInfo& metaInfo,
  ara::diag::CancellationHandler cancellationHandler
) = 0;
```

⌋

8.29.2.2.3.5 StopOffer

[SWS_DM_00595] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::StopOffer

Upstream requirements: RS_Diag_04169, RS_Main_01002

⌈

Kind:	function
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}
Syntax:	void StopOffer () noexcept;
Return value:	None
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	This StopOffer will disable the forwarding of request messages from DM

⌋

8.29.3 Struct: {<routine-interface-name-upper-camel>}RequestResultOutput

[SWS_DM_00583] Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}RequestResultOutput

Status: DRAFT
Upstream requirements: RS_Diag_04169, RS_Diag_04224, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	struct	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Forwarding header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Symbol:	{<routine-interface-name-upper-camel>}RequestResultOutput	
Syntax:	struct {<routine-interface-name-upper-camel>}RequestResultOutput {...};	
Description:	Structure wrapping the *out* arguments for a RequestResult	
Descriptors:	{<routine-interface-name-upper-camel>}	As per {<routine-interface-name-upper-camel>} in [SWS_DM_00591]
	{<routine-interface-request-result-out-args-members-ordered>}	**Shown as "..." in Syntax**. Each element is an argument in the ordered list of arguments, with direction == out **or** direction == inout, where [SWS_DM_80005] applies for each.

⌋

8.29.4 Struct: {<routine-interface-name-upper-camel>}StartOutput

[SWS_DM_00581] Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}StartOutput

Status: DRAFT
Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	struct
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"

▽

△

Forwarding header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Symbol:	{<routine-interface-name-upper-camel>}StartOutput	
Syntax:	struct {<routine-interface-name-upper-camel>}StartOutput {...};	
Description:	Structure wrapping the *out* arguments for a Start	
Descriptors:	{<routine-interface-name-upper-camel>}	As per {<routine-interface-name-upper-camel>} in [SWS_DM_00591]
	{<routine-interface-start-out-args-members-ordered>}	**Shown as "..." in Syntax**. Each element is an argument in the ordered list of arguments, with direction == out **or** direction == inout, where [SWS_DM_80003] applies for each.

⌋

8.29.5 Struct: {<routine-interface-name-upper-camel>}StopOutput

[SWS_DM_00582] Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}StopOutput

Status: DRAFT
Upstream requirements: RS_Diag_04169, RS_Diag_04224, RS_AP_00127, RS_AP_00128, RS_AP_00138

⌈

Kind:	struct	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Forwarding header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}	
Symbol:	{<routine-interface-name-upper-camel>}StopOutput	
Syntax:	struct {<routine-interface-name-upper-camel>}StopOutput {...};	
Description:	Structure wrapping the *out* arguments for a Stop	
Descriptors:	{<routine-interface-name-upper-camel>}	As per {<routine-interface-name-upper-camel>} in [SWS_DM_00591]
	{<routine-interface-stop-out-args-members-ordered>}	**Shown as "..." in Syntax**. Each element is an argument in the ordered list of arguments, with direction == out **or** direction == inout, where [SWS_DM_80004] applies for each.

⌋

8.30 Diagnostic DoIP-related APIs

8.31 Header: ara/diag/doip_activationline.h

8.31.1 Class: DoIPActivationLine

[SWS_DM_00830] Definition of API class ara::diag::DoIPActivationLine

Upstream requirements: RS_Diag_04242

⌈

Kind:	class
Header file:	#include "ara/diag/doip_activationline.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DoIPActivationLine
Syntax:	`class DoIPActivationLine {...};`
Description:	DiagnosticDoIPActivationLineInterface.

⌋

8.31.1.1 Public Member Functions

8.31.1.1.1 Special Member Functions

8.31.1.1.1.1 Move Constructor

[SWS_DM_01634] Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`DoIPActivationLine (DoIPActivationLine &&) noexcept=delete;`
Description:	Move constructor of DoIPActivationLine.

⌋

8.31.1.1.1.2 Move Assignment Operator

[SWS_DM_01632] Definition of API function ara::diag::DoIPActivationLine::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`DoIPActivationLine & operator= (DoIPActivationLine &&)=delete;`
Description:	Move assignment operator of DoIPActivationLine.

⌋

8.31.1.1.1.3 Destructor

[SWS_DM_00832] Definition of API function ara::diag::DoIPActivationLine::~DoIPActivationLine

Upstream requirements: RS_Diag_04242, RS_AP_00134

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`virtual ~DoIPActivationLine () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPActivationLine .

⌋

8.31.1.1.2 Constructors

8.31.1.1.2.1 DoIPActivationLine

[SWS_DM_00831] Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	`class ara::diag::DoIPActivationLine`	
Syntax:	`explicit DoIPActivationLine (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an `DiagnosticDoIPActivationLineInterface`
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticDoIpActivationLinePortMapping` needs to be typed by a `DiagnosticDoIPActivationLineInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of DoIPActivationLine.	

⌋

8.31.1.1.2.2 DoIPActivationLine

[SWS_DM_01633] Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`DoIPActivationLine (DoIPActivationLine &)=delete;`
Description:	DoIPActivationLine shall be a single not copy-able instance.

⌋

8.31.1.1.3 Member Functions

8.31.1.1.3.1 GetActivationLineState

[SWS_DM_00835] Definition of API function ara::diag::DoIPActivationLine::GetActivationLineState

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	`class ara::diag::DoIPActivationLine`	
Syntax:	`virtual ara::core::Future< bool > GetActivationLineState () noexcept=0;`	
Return value:	ara::core::Future< bool >	TRUE in case the activation line is active, else FALSE.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Called to get the current activation line state.	

⌋

8.31.1.1.3.2 GetNetworkInterfaceId

[SWS_DM_00833] Definition of API function ara::diag::DoIPActivationLine::GetNetworkInterfaceId

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	`class ara::diag::DoIPActivationLine`	
Syntax:	`virtual ara::core::Future< std::uint8_t > GetNetworkInterfaceId () noexcept=0;`	
Return value:	ara::core::Future< std::uint8_t >	network interface id for which this activation line is responsible.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Called to get the network interface Id (see DoIpNetworkConfiguration.networkInterfaceId) for which this DoIPActivationLine instance is responsible.	
Notes:	If the reported DoIpNetworkConfiguration.networkInterfaceId belongs to a DoIpNetwork Configuration with property isActivationLineDependent = 'FALSE', this is an error!	

⌋

8.31.1.1.3.3 Offer

[SWS_DM_00836] Definition of API function ara::diag::DoIPActivationLine::Offer

Upstream requirements: RS_Diag_04242, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	`class ara::diag::DoIPActivationLine`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics This service was already offered.
Description:	This Offer will enable the DM to listen to activation line state changes for the given interface.	

⌋

8.31.1.1.3.4 StopOffer

[SWS_DM_00837] Definition of API function ara::diag::DoIPActivationLine::StopOffer

Upstream requirements: RS_Diag_04242

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the provision of activation line state to DM. .

⌋

8.31.1.1.3.5 UpdateActivationLineState

[SWS_DM_00834] Definition of API function ara::diag::DoIPActivationLine::UpdateActivationLineState

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	`class ara::diag::DoIPActivationLine`	
Syntax:	`virtual void UpdateActivationLineState (bool) noexcept=0;`	
DIRECTION NOT DEFINED	bool	--
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Called to update current activation line state. .	

⌋

8.31.1.1.3.6 operator=

[SWS_DM_01631] Definition of API function ara::diag::DoIPActivationLine::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	`class ara::diag::DoIPActivationLine`
Syntax:	`DoIPActivationLine & operator= (DoIPActivationLine &)=delete;`
Description:	DoIPActivationLine shall be a single not assignable instance.

⌋

8.32 Header: ara/diag/doip_entity_identification.h

8.32.1 Class: DoIPEntityIdentification

[SWS_DM_01362] Definition of API class ara::diag::DoIPEntityIdentification

Upstream requirements: RS_Diag_00026

⌈

Kind:	class
Header file:	#include "ara/diag/doip_entity_identification.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DoIPEntityIdentification
Syntax:	`class DoIPEntityIdentification {...};`
Description:	DoIPEntityIdentification class to obtain EID if required.

⌋

8.32.1.1 Public Member Functions

8.32.1.1.1 Special Member Functions

8.32.1.1.1.1 Destructor

[SWS_DM_01365] Definition of API function ara::diag::DoIPEntityIdentification::~DoIPEntityIdentification

Upstream requirements: RS_AP_00134, RS_Diag_00026

⌈

Kind:	function
Header file:	#include "ara/diag/doip_entity_identification.h"
Scope:	`class ara::diag::DoIPEntityIdentification`
Syntax:	`virtual ~DoIPEntityIdentification () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPEntityIdentification. .

⌋

8.32.1.1.2 Constructors

8.32.1.1.2.1 DoIPEntityIdentification

[SWS_DM_01364] Definition of API function ara::diag::DoIPEntityIdentification::DoIPEntityIdentification

Upstream requirements: RS_AP_00137, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_entity_identification.h"	
Scope:	`class ara::diag::DoIPEntityIdentification`	
Syntax:	`explicit DoIPEntityIdentification (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPEntity IdentificationInterface.
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticDoIpEntityIdentificationPortMapping` needs to be typed by a `DiagnosticDoIPEntityIdentificationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of DoIPEntityIdentification.	

⌋

8.32.1.1.3 Member Functions

8.32.1.1.3.1 GetEntityId

[SWS_DM_01366] Definition of API function ara::diag::DoIPEntityIdentification::GetEntityId

Upstream requirements: RS_AP_00138, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_entity_identification.h"	
Scope:	`class ara::diag::DoIPEntityIdentification`	
Syntax:	`virtual ara::core::Future< EntityId > GetEntityId () noexcept=0;`	
Return value:	ara::core::Future< EntityId >	Entity identification
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called to get the EID for the DoIP protocol.	

⌋

8.32.1.1.3.2 Offer

[SWS_DM_01367] Definition of API function ara::diag::DoIPEntityIdentification::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_entity_identification.h"	
Scope:	`class ara::diag::DoIPEntityIdentification`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.32.1.1.3.3 StopOffer

[SWS_DM_01368] Definition of API function ara::diag::DoIPEntityIdentification::StopOffer

Upstream requirements: RS_Diag_00026

⌈

Kind:	function
Header file:	#include "ara/diag/doip_entity_identification.h"
Scope:	`class ara::diag::DoIPEntityIdentification`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM. .

⌋

8.32.2 Struct: EntityId

[SWS_DM_01363] Definition of API class ara::diag::DoIPEntityIdentification::EntityId

Upstream requirements: RS_Diag_00026

⌈

Kind:	struct
Header file:	#include "ara/diag/doip_entity_identification.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DoIPEntityIdentification`
Symbol:	EntityId
Syntax:	`struct EntityId {...};`
Description:	Entity Identification value as defined in ISO13400-2.

⌋

8.32.2.1 Public Member Variables

8.32.2.1.1 entityIdentification

[SWS_DM_01496] Definition of API variable ara::diag::DoIPEntityIdentification::EntityId::entityIdentification

Upstream requirements: RS_Diag_00026

⌈

Kind:	variable
Header file:	#include "ara/diag/doip_entity_identification.h"
Scope:	`struct ara::diag::DoIPEntityIdentification::EntityId`
Symbol:	entityIdentification
Type:	`ara::core::Array< std::uint8_t, 6 >`
Syntax:	`ara::core::Array<std::uint8_t, 6> entityIdentification;`
Description:	Entity Identification value.

⌋

8.33 Header: ara/diag/doip_group_identification.h

8.33.1 Class: DoIPGroupIdentification

[SWS_DM_00720] Definition of API class ara::diag::DoIPGroupIdentification

Upstream requirements: RS_Diag_00026

⌈

Kind:	class
Header file:	#include "ara/diag/doip_group_identification.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DoIPGroupIdentification
Syntax:	`class DoIPGroupIdentification {...};`
Description:	DoIPGroupIdentificationInterface.

⌋

8.33.1.1 Public Member Functions

8.33.1.1.1 Special Member Functions

8.33.1.1.1.1 Move Constructor

[SWS_DM_01642] Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`DoIPGroupIdentification (DoIPGroupIdentification &&) noexcept=delete;`
Description:	Move constructor of DoIPGroupIdentification.

⌋

8.33.1.1.1.2 Move Assignment Operator

[SWS_DM_01640] Definition of API function ara::diag::DoIPGroupIdentification::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`DoIPGroupIdentification & operator= (DoIPGroupIdentification &&)=delete;`
Description:	Move assignment operator of DoIPGroupIdentification.

⌋

8.33.1.1.1.3 Destructor

[SWS_DM_00723] Definition of API function ara::diag::DoIPGroupIdentification::~DoIPGroupIdentification

Upstream requirements: RS_AP_00134, RS_Diag_00026

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`virtual ~DoIPGroupIdentification () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPGroupIdentification .

⌋

8.33.1.1.2 Constructors

8.33.1.1.2.1 DoIPGroupIdentification

[SWS_DM_00722] Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification

Upstream requirements: RS_AP_00137, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_group_identification.h"	
Scope:	`class ara::diag::DoIPGroupIdentification`	
Syntax:	`explicit DoIPGroupIdentification (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPGroup IdentificationInterface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticDoIpGroupIdentificationPortMapping` needs to be typed by a `DiagnosticDoIPGroupIdentificationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of DoIPGroupIdentification.	

⌋

8.33.1.1.2.2 DoIPGroupIdentification

[SWS_DM_01641] Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`DoIPGroupIdentification (DoIPGroupIdentification &)=delete;`
Description:	DoIPGroupIdentification shall be a single not copy-able instance.

⌋

8.33.1.1.3 Member Functions

8.33.1.1.3.1 GetGidStatus

[SWS_DM_00724] Definition of API function ara::diag::DoIPGroupIdentification::GetGidStatus

Upstream requirements: RS_AP_00138, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_group_identification.h"	
Scope:	`class ara::diag::DoIPGroupIdentification`	
Syntax:	`virtual ara::core::Future< GidStatus > GetGidStatus () noexcept=0;`	
Return value:	ara::core::Future< Gid Status >	group identification and state
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called to get the current GID state for the DoIP protocol.	

⌋

8.33.1.1.3.2 Offer

[SWS_DM_00725] Definition of API function ara::diag::DoIPGroupIdentification::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_00026

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_group_identification.h"	
Scope:	`class ara::diag::DoIPGroupIdentification`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.33.1.1.3.3 StopOffer

[SWS_DM_00726] Definition of API function ara::diag::DoIPGroupIdentification::StopOffer

Upstream requirements: RS_Diag_00026

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.33.1.1.3.4 operator=

[SWS_DM_01639] Definition of API function ara::diag::DoIPGroupIdentification::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_group_identification.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Syntax:	`DoIPGroupIdentification & operator= (DoIPGroupIdentification &)=delete;`
Description:	DoIPGroupIdentification shall be a single not assignable instance.

⌋

8.33.2 Struct: GidStatus

[SWS_DM_00721] Definition of API class ara::diag::DoIPGroupIdentification::GidStatus

Upstream requirements: RS_Diag_00026

⌈

Kind:	struct
Header file:	#include "ara/diag/doip_group_identification.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DoIPGroupIdentification`
Symbol:	GidStatus
Syntax:	`struct GidStatus {...};`
Description:	Response data of positive response message.

⌋

8.34 Header: ara/diag/doip_power_mode.h

8.34.1 Non-Member Types

8.34.1.1 Enumeration: PowerModeType

[SWS_DM_00730] Definition of API enum ara::diag::PowerModeType

Upstream requirements: RS_Diag_00080

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/doip_power_mode.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	PowerModeType	
Underlying type:	--	
Syntax:	`enum class PowerModeType {...};`	
Values:	kNotReady= 0x00	not all ECUs accessible via DoIP can communicate
	kReady= 0x01	all ECUs accessible via DoIP can communicate
	kNotSupported= 0x02	the Diagnostic Information Power Mode Information Request message is not supported
Description:	PowerMode as defined in ISO13400-2.	

⌋

8.34.2 Class: DoIPPowerMode

[SWS_DM_00731] Definition of API class ara::diag::DoIPPowerMode

Upstream requirements: RS_Diag_00080

⌈

Kind:	class
Header file:	#include "ara/diag/doip_power_mode.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DoIPPowerMode
Syntax:	`class DoIPPowerMode {...};`
Description:	DiagnosticDoIPPowerModeInterface.

⌋

8.34.2.1 Public Member Functions

8.34.2.1.1 Special Member Functions

8.34.2.1.1.1 Move Constructor

[SWS_DM_01646] Definition of API function ara::diag::DoIPPowerMode::DoIP-PowerMode

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	`class ara::diag::DoIPPowerMode`
Syntax:	`DoIPPowerMode (DoIPPowerMode &&) noexcept=delete;`
Description:	Move constructor of DoIPPowerMode.

⌋

8.34.2.1.1.2 Move Assignment Operator

[SWS_DM_01644] Definition of API function ara::diag::DoIPPower Mode::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	`class ara::diag::DoIPPowerMode`
Syntax:	`DoIPPowerMode & operator= (DoIPPowerMode &&)=delete;`
Description:	Move assignment operator of DoIPPowerMode.

⌋

8.34.2.1.1.3 Destructor

[SWS_DM_00733] Definition of API function ara::diag::DoIPPowerMode::~DoIP-PowerMode

Upstream requirements: RS_AP_00134, RS_Diag_00080

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	`class ara::diag::DoIPPowerMode`
Syntax:	`virtual ~DoIPPowerMode () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPPowerMode .

⌋

8.34.2.1.2 Constructors

8.34.2.1.2.1 DoIPPowerMode

[SWS_DM_00732] Definition of API function ara::diag::DoIPPowerMode::DoIP-PowerMode

Upstream requirements: RS_AP_00137, RS_Diag_00080

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_power_mode.h"	
Scope:	`class ara::diag::DoIPPowerMode`	
Syntax:	`explicit DoIPPowerMode (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPPower ModeInterface
	concurrencyType	Specifies if the interface is implemented as thread-safe(k Concurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpeci-fierMappingIn-tegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	PortInterfaceMappingViolation	A PortPrototype that is referenced by a DiagnosticDoIpPowerModePortMapping needs to be typed by a DiagnosticDoIPPowerModeInterface.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of DoIPPowerMode.	

⌋

8.34.2.1.2.2 DoIPPowerMode

[SWS_DM_01645] Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	class ara::diag::DoIPPowerMode
Syntax:	DoIPPowerMode (DoIPPowerMode &)=delete;
Description:	DoIPPowerMode shall be a single not copy-able instance.

⌋

8.34.2.1.3 Member Functions

8.34.2.1.3.1 GetDoIPPowerMode

[SWS_DM_00734] Definition of API function ara::diag::DoIPPowerMode::GetDoIPPowerMode

Upstream requirements: RS_AP_00138, RS_Diag_00080

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_power_mode.h"	
Scope:	`class ara::diag::DoIPPowerMode`	
Syntax:	`virtual ara::core::Future< PowerModeType > GetDoIPPowerMode () noexcept=0;`	
Return value:	ara::core::Future< PowerModeType >	current diagnostic power mode
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called to get the current Power Mode for the DoIP protocol.	

⌋

8.34.2.1.3.2 Offer

[SWS_DM_00735] Definition of API function ara::diag::DoIPPowerMode::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_00080

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_power_mode.h"	
Scope:	`class ara::diag::DoIPPowerMode`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.34.2.1.3.3 StopOffer

[SWS_DM_00736] Definition of API function ara::diag::DoIPPowerMode::StopOffer

Upstream requirements: RS_Diag_00080

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	`class ara::diag::DoIPPowerMode`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.34.2.1.3.4 operator=

[SWS_DM_01643] Definition of API function ara::diag::DoIPPowerMode::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	`class ara::diag::DoIPPowerMode`
Syntax:	`DoIPPowerMode & operator= (DoIPPowerMode &)=delete;`
Description:	DoIPPowerMode shall be a single not assignable instance.

⌋

8.35 Header: ara/diag/doip_trigger_vehicle_announcement.h

8.35.1 Class: DoIPTriggerVehicleAnnouncement

[SWS_DM_00820] Definition of API class ara::diag::DoIPTriggerVehicleAnnouncement

Upstream requirements: RS_Diag_04242

⌈

Kind:	class
Header file:	#include "ara/diag/doip_trigger_vehicle_announcement.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DoIPTriggerVehicleAnnouncement
Syntax:	`class DoIPTriggerVehicleAnnouncement {...};`
Description:	DiagnosticDoIPTriggerVehicleAnnouncement.

⌋

8.35.1.1 Public Member Functions

8.35.1.1.1 Member Functions

8.35.1.1.1.1 GetDoIPTriggerVehicleAnnouncement

[SWS_DM_00821] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::GetDoIPTriggerVehicleAnnouncement

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_trigger_vehicle_announcement.h"	
Scope:	`class ara::diag::DoIPTriggerVehicleAnnouncement`	
Syntax:	`static ara::core::Result< std::reference_wrapper< DoIPTriggerVehicle Announcement > > GetDoIPTriggerVehicleAnnouncement () noexcept;`	
Return value:	ara::core::Result< std::reference_wrapper< DoIPTriggerVehicle Announcement > >	DoIPTriggerVehicleAnnouncement object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Get DoIPTriggerVehicleAnnouncement interface from DM.	

⌋

8.35.1.1.1.2 TriggerVehicleAnnouncement

[SWS_DM_00822] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement

Upstream requirements: RS_Diag_04242

⌈

Kind:	function	
Header file:	#include "ara/diag/doip_trigger_vehicle_announcement.h"	
Scope:	`class ara::diag::DoIPTriggerVehicleAnnouncement`	
Syntax:	`virtual ara::core::Result< void > TriggerVehicleAnnouncement (std::uint8_t networkInterfaceId) noexcept=0;`	
DIRECTION NOT DEFINED	networkInterfaceId	--
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Called by application to trigger DM sending out vehicle announcements on the given network interface Id.	
Notes:	If the reported DoIPNetworkConfiguration.networkInterfaceId belongs to a DoIPNetwork Configuration with property isActivationLineDependent = 'TRUE', this is an error as on those interfaces sending of announcements happens automatically after activation line going up/ip address assignment.	

⌋

8.35.1.2 Private Member Functions

8.35.1.2.1 Special Member Functions

8.35.1.2.1.1 Default Constructor

[SWS_DM_02100] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::DoIPTriggerVehicleAnnouncement ⌈

Kind:	function
Header file:	#include "ara/diag/doip_trigger_vehicle_announcement.h"
Scope:	`class ara::diag::DoIPTriggerVehicleAnnouncement`
Syntax:	`DoIPTriggerVehicleAnnouncement ()=delete;`
Description:	Ctor is vendor-specific.
Visibility:	private

⌋

8.35.1.2.1.2 Destructor

[SWS_DM_02101] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::~DoIPTriggerVehicleAnnouncement ⌈

Kind:	function
Header file:	#include "ara/diag/doip_trigger_vehicle_announcement.h"
Scope:	`class ara::diag::DoIPTriggerVehicleAnnouncement`
Syntax:	`~DoIPTriggerVehicleAnnouncement () noexcept=delete;`
Description:	Dtor is vendor-specific.
Visibility:	private

⌋

8.36 Diagnostic Event-related APIs

8.37 Header: ara/diag/condition.h

8.37.1 Non-Member Types

8.37.1.1 Enumeration: ConditionType

[SWS_DM_00710] Definition of API enum ara::diag::ConditionType

Upstream requirements: RS_AP_00125, RS_Diag_04192

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/condition.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	ConditionType	
Underlying type:	--	
Syntax:	`enum class ConditionType {...};`	
Values:	kConditionFalse= 0x00	condition is set to false
	kConditionTrue= 0x01	condition is set to true
Description:	Type for Condition status.	

⌋

8.37.2 Class: Condition

[SWS_DM_00711] Definition of API class ara::diag::Condition

Upstream requirements: RS_Diag_04192

⌈

Kind:	class
Header file:	#include "ara/diag/condition.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Condition
Syntax:	`class Condition final {...};`
Description:	DiagnosticConditionInterface.

⌋

8.37.2.1 Public Member Functions

8.37.2.1.1 Special Member Functions

8.37.2.1.1.1 Move Constructor

[SWS_DM_01626] Definition of API function ara::diag::Condition::Condition

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`
Syntax:	`Condition (Condition &&) noexcept=delete;`
Description:	Move constructor of Condition.

⌋

8.37.2.1.1.2 Move Assignment Operator

[SWS_DM_01624] Definition of API function ara::diag::Condition::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`
Syntax:	`Condition & operator= (Condition &&)=delete;`
Description:	Move assignment operator of Condition.

⌋

8.37.2.1.1.3 Destructor

[SWS_DM_00713] Definition of API function ara::diag::Condition::~Condition

Upstream requirements: RS_AP_00134, RS_Diag_04192

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`
Syntax:	`~Condition () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class Condition .

⌋

8.37.2.1.2 Constructors

8.37.2.1.2.1 Condition

[SWS_DM_00712] Definition of API function ara::diag::Condition::Condition

Upstream requirements: RS_AP_00137, RS_Diag_04192

⌈

Kind:	function	
Header file:	#include "ara/diag/condition.h"	
Scope:	`class ara::diag::Condition`	
Syntax:	`explicit Condition (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticCondition Interface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is typed by a `DiagnosticConditionInterface` needs either to be referenced by a `DiagnosticEnableConditionPortMapping` or a `DiagnosticClearConditionPortMapping`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	s
Description:	Constructor of Condition Class.	

⌋

8.37.2.1.2.2 Condition

[SWS_DM_01625] Definition of API function ara::diag::Condition::Condition

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`

▽

△

Syntax:	`Condition (Condition &)=delete;`
Description:	Condition shall be a single not copy-able instance.

⌋

8.37.2.1.3 Member Functions

8.37.2.1.3.1 GetCondition

[SWS_DM_00714] Definition of API function ara::diag::Condition::GetCondition

Upstream requirements: RS_AP_00139, RS_Diag_04192

⌈

Kind:	function	
Header file:	#include "ara/diag/condition.h"	
Scope:	`class ara::diag::Condition`	
Syntax:	`ara::core::Future< ConditionType > GetCondition () noexcept;`	
Return value:	ara::core::Future< ConditionType >	the current condition
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get current condition.	

⌋

8.37.2.1.3.2 SetCondition

[SWS_DM_00715] Definition of API function ara::diag::Condition::SetCondition

Upstream requirements: RS_AP_00139, RS_Diag_04192

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`

▽

△

Syntax:	`ara::core::Result< void > SetCondition (ConditionType condition) noexcept;`	
Parameters (in):	condition	current condition
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Set condition.	

⌋

8.37.2.1.3.3 operator=

[SWS_DM_01623] Definition of API function ara::diag::Condition::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	`class ara::diag::Condition`
Syntax:	`Condition & operator= (Condition &)=delete;`
Description:	Condition shall be a single not assignable instance.

⌋

8.38 Header: ara/diag/operation_cycle.h

8.38.1 Class: OperationCycle

[SWS_DM_00751] Definition of API class ara::diag::OperationCycle

Upstream requirements: RS_Diag_04178

⌈

Kind:	class
Header file:	#include "ara/diag/operation_cycle.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	OperationCycle
Syntax:	`class OperationCycle final {...};`

▽

△

Description:	DiagnosticOperationCycleInterface provides functionality for handling of operation cycles.

⌋

8.38.1.1 Public Member Types

8.38.1.1.1 Type Alias: OperationCycleSetNotifier

[SWS_DM_02078] Definition of API type ara::diag::OperationCycle::OperationCycleSetNotifier

Upstream requirements: RS_AP_00139, RS_Diag_04178, RS_Diag_04186

⌈

Kind:	type alias
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`
Symbol:	OperationCycleSetNotifier
Syntax:	`using OperationCycleSetNotifier = std::function<void(void)>;`
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the operation cycle is changed. .

⌋

8.38.1.2 Public Member Functions

8.38.1.2.1 Special Member Functions

8.38.1.2.1.1 Move Constructor

[SWS_DM_01599] Definition of API function ara::diag::OperationCycle::OperationCycle

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`

▽

△

Syntax:	`OperationCycle (OperationCycle &&) noexcept=delete;`
Description:	Move constructor of OperationCycle.

⌋

8.38.1.2.1.2 Move Assignment Operator

[SWS_DM_01597] Definition of API function ara::diag::OperationCycle::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`
Syntax:	`OperationCycle & operator= (OperationCycle &&)=delete;`
Description:	Move assignment operator of OperationCycle.

⌋

8.38.1.2.1.3 Destructor

[SWS_DM_00753] Definition of API function ara::diag::OperationCycle::~OperationCycle

Upstream requirements: RS_AP_00134, RS_Diag_04178

⌈

Kind:	function
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`
Syntax:	`~OperationCycle () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticOperationCycleInterface .

⌋

8.38.1.2.2 Constructors

8.38.1.2.2.1 OperationCycle

[SWS_DM_00752] Definition of API function ara::diag::OperationCycle::OperationCycle

Upstream requirements: RS_AP_00137, RS_Diag_04178

⌈

Kind:	function	
Header file:	#include "ara/diag/operation_cycle.h"	
Scope:	`class ara::diag::OperationCycle`	
Syntax:	`explicit OperationCycle (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticOperationCycleInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticOperationCyclePortMapping` needs to be typed by a `DiagnosticOperationCycleInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for DiagnosticOperationCycleInterface.	

⌋

8.38.1.2.2.2 OperationCycle

[SWS_DM_01598] Definition of API function ara::diag::OperationCycle::OperationCycle

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`
Syntax:	`OperationCycle (OperationCycle &)=delete;`
Description:	OperationCycle shall be a single not copy-able instance.

⌋

8.38.1.2.3 Member Functions

8.38.1.2.3.1 RestartOperationCycle

[SWS_DM_01102] Definition of API function ara::diag::OperationCycle::RestartOperationCycle

Upstream requirements: RS_AP_00139, RS_Diag_04178, RS_Diag_04182

⌈

Kind:	function	
Header file:	#include "ara/diag/operation_cycle.h"	
Scope:	`class ara::diag::OperationCycle`	
Syntax:	`ara::core::Result< void > RestartOperationCycle () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Trigger to restart the OperationCycle .	

⌋

8.38.1.2.3.2 SetNotifier

[SWS_DM_00755] Definition of API function ara::diag::OperationCycle::SetNotifier

Upstream requirements: RS_AP_00139, RS_Diag_04178, RS_Diag_04186

⌈

Kind:	function	
Header file:	#include "ara/diag/operation_cycle.h"	
Scope:	`class ara::diag::OperationCycle`	
Syntax:	`ara::core::Result< void > SetNotifier (OperationCycleSetNotifier notifier) noexcept;`	
DIRECTION NOT DEFINED	notifier	--
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Registering a notifier function which is called if the operation cycle is changed. A consecutive call of this method will overwrite the previous registered notifier. .	

⌋

8.38.1.2.3.3 operator=

[SWS_DM_01596] Definition of API function ara::diag::OperationCycle::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/operation_cycle.h"
Scope:	`class ara::diag::OperationCycle`
Syntax:	`OperationCycle & operator= (OperationCycle &)=delete;`
Description:	OperationCycle shall be a single not assignable instance.

⌋

8.39 Header: ara/diag/monitor.h

8.39.1 Class: Monitor

[SWS_DM_00542] Definition of API class ara::diag::Monitor

Upstream requirements: RS_Diag_04179

⌈

Kind:	class
Header file:	#include "ara/diag/monitor.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Monitor
Syntax:	`class Monitor final {...};`
Description:	Class to implement operations on diagnostic Monitor interface.

⌋

8.39.1.1 Public Member Functions

8.39.1.1.1 Special Member Functions

8.39.1.1.1.1 Move Constructor

[SWS_DM_01686] Definition of API function ara::diag::Monitor::Monitor

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	`class ara::diag::Monitor`
Syntax:	`Monitor (Monitor &&) noexcept=delete;`
Description:	Move constructor of Monitor.

⌋

8.39.1.1.1.2 Move Assignment Operator

[SWS_DM_01684] Definition of API function ara::diag::Monitor::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	`class ara::diag::Monitor`
Syntax:	`Monitor & operator= (Monitor &&)=delete;`
Description:	Move assignment operator of Monitor.

⌋

8.39.1.1.2 Constructors

8.39.1.1.2.1 Monitor

[SWS_DM_01972] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	`class ara::diag::Monitor`	
Syntax:	`Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMonitorPortMapping` needs to be typed by a `DiagnosticMonitorInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	InstanceSpecifier-AlreadyInUseViola-tion	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
	InvalidDebouncin-gAlgorithm	Mismatching Debounce Algorithm in DEXT i.e DiagnosticEventToDebounceAlgorithmMapping shall be mapped to DiagnosticEventToDebounceAlgorithmMapping. DiagnosticDebounceAlgorithmProps which aggregates DiagEventDebounceMonitorInternal or DiagnosticEventToDebounceAlgorithmMapping shall not exist.
Description:	Monitor constructor for Monitors without debouncing.	

⌋

8.39.1.1.2.2 Monitor

[SWS_DM_00549] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_Diag_04068, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	class ara::diag::Monitor	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	InstanceSpeci-fierMappingIn-tegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMap-pingViolation	A PortPrototype that is referenced by a DiagnosticMonitorPortMapping needs to be typed by a DiagnosticMonitorInterface.
	ProcessMappingVio-lation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	InstanceSpecifier-AlreadyInUseViola-tion	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
	InvalidDebouncin-gAlgorithm	Mismatching Debounce Algorithm in DEXT i.e DiagnosticEventToDebounceAlgorithmMapping shall be mapped to DiagnosticEventToDebounceAlgorithmMapping. DiagnosticDebounceAlgorithmProps which aggregates DiagEventDebounceCounterBased.
Description:	Monitor constructor for Monitors with counter-based debouncing.	

⌋

8.39.1.1.2.3 Monitor

[SWS_DM_00550] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_Diag_04225, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	class ara::diag::Monitor	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, TimeBased defaultValues) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	InstanceSpeci-fierMappingIn-tegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMap-pingViolation	A PortPrototype that is referenced by a DiagnosticMonitorPortMapping needs to be typed by a DiagnosticMonitorInterface.
	ProcessMappingVio-lation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	InstanceSpecifier-AlreadyInUseViola-tion	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
	InvalidDebouncin-gAlgorithm	Mismatching Debounce Algorithm in DEXT i.e DiagnosticEventToDebounceAlgorithmMapping shall be mapped to DiagnosticEventToDebounceAlgorithmMapping. DiagnosticDebounceAlgorithmProps which aggregates DiagEventDebounceTimeBased.
Description:	Monitor constructor for Monitors with time-based debouncing.	

⌋

8.39.1.1.2.4 Monitor

[SWS_DM_01685] Definition of API function ara::diag::Monitor::Monitor

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	class ara::diag::Monitor
Syntax:	Monitor (Monitor &)=delete;
Description:	Monitor shall be a single not copy-able instance.

⌋

8.39.1.1.2.5 Monitor

[SWS_DM_00548] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_AP_00121

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	class ara::diag::Monitor

▽

△

Syntax:	`Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetection Counter	Possibility to register a function to get the current FDC for this event.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMonitorPortMapping` needs to be typed by a `DiagnosticMonitorInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
	`InvalidDebouncingAlgorithm`	Mismatching Debounce Algorithm in DEXT i.e `DiagnosticEventToDebounceAlgorithmMapping` shall be mapped to `DiagnosticEventToDebounceAlgorithmMapping`. `DiagnosticDebounceAlgorithmProps` which aggregates `DiagEventDebounceMonitorInternal` or `DiagnosticEventToDebounceAlgorithmMapping` shall not exist.
Description:	Monitor constructor for Monitors with Monitor-internal debouncing. using ara::diag::CounterBased; using ara::diag::TimeBased;	

⌋

8.39.1.1.3 Member Functions

8.39.1.1.3.1 Offer

[SWS_DM_01088] Definition of API function ara::diag::Monitor::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	`class ara::diag::Monitor`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	With the Offer the application states that it is ready to receive and process initMonitor callouts.	

⌋

8.39.1.1.3.2 ReportMonitorAction

[SWS_DM_00543] Definition of API function ara::diag::Monitor::ReportMonitor Action

Upstream requirements: RS_Diag_04179, RS_AP_00139, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	`class ara::diag::Monitor`	
Syntax:	`void ReportMonitorAction (MonitorAction action) noexcept;`	
Parameters (in):	action	Contains either the last (un-)qualified test result of the diagnostic monitor or commands to control the debouncing or to force a prestorage.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Violations:	`UnexpectedMoni-torActionHandling`	`ara::diag::Monitor::ReportMonitorAction` is called with a parameter `action` value that is not supported by the used debouncing method.
Description:	Function to report the status information being relevant for error monitoring paths.	

⌋

8.39.1.1.3.3 StopOffer

[SWS_DM_01089] Definition of API function ara::diag::Monitor::StopOffer

Upstream requirements: RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	`class ara::diag::Monitor`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.39.1.1.3.4 operator=

[SWS_DM_01683] Definition of API function ara::diag::Monitor::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/monitor.h"
Scope:	`class ara::diag::Monitor`
Syntax:	`Monitor & operator= (Monitor &)=delete;`
Description:	Monitor shall be a single not assignable instance.

⌋

8.40 Header: ara/diag/monitor_types.h

8.40.1 Non-Member Types

8.40.1.1 Enumeration: InitMonitorReason

[SWS_DM_00540] Definition of API enum ara::diag::InitMonitorReason

Upstream requirements: RS_Diag_04179

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/monitor_types.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	InitMonitorReason	
Underlying type:	std::uint32_t	
Syntax:	`enum class InitMonitorReason : std::uint32_t {...};`	
Values:	kClear= 0x00	Event was cleared and all internal values and states are reset.
	kRestart= 0x01	Operation cycle of the event was (re-)started.
	kReenabled= 0x02	Enable conditions are fulfilled and control DTC setting is set to on
	kDisabled= 0x03	Enable conditions no longer fulfilled, or Control DTC setting is set to off
Description:	Represents the status information reported to AAs why the monitor may be re-initalized.	

⌋

8.40.1.2 Enumeration: MonitorAction

[SWS_DM_00541] Definition of API enum ara::diag::MonitorAction

Upstream requirements: RS_Diag_04179

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/monitor_types.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	MonitorAction	
Underlying type:	std::uint32_t	
Syntax:	`enum class MonitorAction : std::uint32_t {...};`	
Values:	kPassed= 0x00	Monitor reports qualified test result passed.
	kFailed= 0x01	Monitor reports qualified test result failed.

▽

△

	kPrepassed= 0x02	Monitor reports unqualified test result pre-passed.
	kPrefailed= 0x03	Monitor reports unqualified test result pre-failed.
	kFdcThreshold Reached= 0x04	Monitor triggers the storage of ExtendedDataRecords and Freeze Frames (if the triggering condition is connected to this threshold).
	kResetTestFailed= 0x05	Reset TestFailed Bit without any other side effects like readiness
	kFreezeDebouncing= 0x06	Freeze the internal debounce timer.
	kResetDebouncing= 0x07	Reset the internal debounce counter/timer.
Description:	Represents the status information reported by AAs being relevant for error monitoring.	

⌋

8.40.2 Struct: CounterBased

[SWS_DM_00538] Definition of API class ara::diag::CounterBased

Upstream requirements: RS_Diag_04068

⌈

Kind:	struct
Header file:	#include "ara/diag/monitor_types.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	CounterBased
Syntax:	`struct CounterBased {...};`
Description:	Represents the parameters for counter-based debouncing.

⌋

8.40.2.1 Public Member Variables

8.40.2.1.1 failedJumpValue

[SWS_DM_00625] Definition of API variable ara::diag::CounterBased::failedJumpValue

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	failedJumpValue
Type:	`std::int16_t`
Syntax:	`std::int16_t failedJumpValue;`
Description:	failed to jump value

⌋

8.40.2.1.2 failedStepsize

[SWS_DM_00623] Definition of API variable ara::diag::CounterBased::failedStepsize

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	failedStepsize
Type:	`std::uint16_t`
Syntax:	`std::uint16_t failedStepsize;`
Description:	Stepsize per pre-failed report.

⌋

8.40.2.1.3 failedThreshold

[SWS_DM_00621] Definition of API variable ara::diag::CounterBased::failedThreshold

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	failedThreshold
Type:	`std::int16_t`
Syntax:	`std::int16_t failedThreshold;`
Description:	Threshold until qualified failed.

⌋

8.40.2.1.4 passedJumpValue

[SWS_DM_00626] Definition of API variable ara::diag::CounterBased::passedJumpValue

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	passedJumpValue
Type:	`std::int16_t`
Syntax:	`std::int16_t passedJumpValue;`
Description:	passed to jump value

⌋

8.40.2.1.5 passedStepsize

[SWS_DM_00624] Definition of API variable ara::diag::CounterBased::passedStepsize

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	passedStepsize
Type:	`std::uint16_t`
Syntax:	`std::uint16_t passedStepsize;`
Description:	Stepsize per pre-passed report.

⌋

8.40.2.1.6 passedThreshold

[SWS_DM_00622] Definition of API variable ara::diag::CounterBased::passedThreshold

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	passedThreshold
Type:	`std::int16_t`
Syntax:	`std::int16_t passedThreshold;`
Description:	Threshold until qualified passed.

⌋

8.40.2.1.7 useJumpToFailed

[SWS_DM_00627] Definition of API variable ara::diag::CounterBased::useJumpToFailed

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	useJumpToFailed
Type:	`bool`
Syntax:	`bool useJumpToFailed;`
Description:	is jump supported

⌋

8.40.2.1.8 useJumpToPassed

[SWS_DM_00628] Definition of API variable ara::diag::CounterBased::useJumpToPassed

Upstream requirements: RS_Diag_04068

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::CounterBased`
Symbol:	useJumpToPassed
Type:	`bool`
Syntax:	`bool useJumpToPassed;`
Description:	is jump supported

⌋

8.40.3 Struct: TimeBased

[SWS_DM_00539] Definition of API class ara::diag::TimeBased

Upstream requirements: RS_Diag_04225

⌈

Kind:	struct
Header file:	#include "ara/diag/monitor_types.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	TimeBased
Syntax:	`struct TimeBased {...};`
Description:	Represents the parameters for time-based debouncing.

⌋

8.40.3.1 Public Member Variables

8.40.3.1.1 failedMs

[SWS_DM_00629] Definition of API variable ara::diag::TimeBased::failedMs

Upstream requirements: RS_Diag_04225

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::TimeBased`
Symbol:	failedMs
Type:	`std::uint32_t`
Syntax:	`std::uint32_t failedMs;`
Description:	time until failed in (ms)

⌋

8.40.3.1.2 passedMs

[SWS_DM_00630] Definition of API variable ara::diag::TimeBased::passedMs

Upstream requirements: RS_Diag_04225

⌈

Kind:	variable
Header file:	#include "ara/diag/monitor_types.h"
Scope:	`struct ara::diag::TimeBased`
Symbol:	passedMs
Type:	`std::uint32_t`
Syntax:	`std::uint32_t passedMs;`
Description:	time until passed in (ms)

⌋

8.41 Header: ara/diag/multiple_condition.h

8.41.1 Non-Member Types

8.41.1.1 Type Alias: ConditionHandleType

[SWS_DM_01726] Definition of API type ara::diag::ConditionHandleType

Upstream requirements: RS_Diag_04192

⌈

Kind:	type alias
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`namespace ara::diag`
Symbol:	ConditionHandleType
Syntax:	`using ConditionHandleType = std::uint8_t;`
Description:	Type to identify a condiition .

⌋

8.41.2 Class: MultipleCondition

[SWS_DM_01730] Definition of API class ara::diag::MultipleCondition

Upstream requirements: RS_Diag_04192

⌈

Kind:	class
Header file:	#include "ara/diag/multiple_condition.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	MultipleCondition
Syntax:	`class MultipleCondition final {...};`
Description:	Class for multiple condition operations.

⌋

8.41.2.1 Public Member Functions

8.41.2.1.1 Special Member Functions

8.41.2.1.1.1 Move Constructor

[SWS_DM_02071] Definition of API function ara::diag::MultipleCondition::MultipleCondition

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`class ara::diag::MultipleCondition`
Syntax:	`MultipleCondition (MultipleCondition &&) noexcept=delete;`
Description:	Move constructor of MultipleCondition.

⌋

8.41.2.1.1.2 Move Assignment Operator

[SWS_DM_02074] Definition of API function ara::diag::MultipleCondition::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`class ara::diag::MultipleCondition`
Syntax:	`MultipleCondition & operator= (MultipleCondition &&)=delete;`
Description:	Move assignment operator of MultipleCondition.

⌋

8.41.2.1.1.3 Destructor

[SWS_DM_01737] Definition of API function ara::diag::MultipleCondition::~MultipleCondition

Upstream requirements: RS_AP_00134, RS_Diag_04192

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`class ara::diag::MultipleCondition`
Syntax:	`~MultipleCondition () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class Condition .

⌋

8.41.2.1.2 Constructors

8.41.2.1.2.1 MultipleCondition

[SWS_DM_01727] Definition of API function ara::diag::MultipleCondition::MultipleCondition

Upstream requirements: RS_AP_00137, RS_Diag_04192

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_condition.h"	
Scope:	`class ara::diag::MultipleCondition`	
Syntax:	`explicit MultipleCondition (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMultipleConditionInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMultipleConditionPortMapping` needs to be typed by a `DiagnosticMultipleConditionInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor of MultipleCondition Class.	

⌋

8.41.2.1.2.2 MultipleCondition

[SWS_DM_02072] Definition of API function ara::diag::MultipleCondition::MultipleCondition

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`class ara::diag::MultipleCondition`
Syntax:	`MultipleCondition (MultipleCondition &)=delete;`
Description:	Copy constructor of MultipleCondition.

⌋

8.41.2.1.3 Member Functions

8.41.2.1.3.1 GetCondition

[SWS_DM_01728] Definition of API function ara::diag::MultipleCondition::GetCondition

Upstream requirements: RS_AP_00139, RS_Diag_04192

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_condition.h"	
Scope:	`class ara::diag::MultipleCondition`	
Syntax:	`ara::core::Future< ConditionType > GetCondition (ConditionHandleType conditionHandle) noexcept;`	
Parameters (in):	conditionHandle	Identifies the requested condition
Return value:	ara::core::Future< ConditionType >	the current condition
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided conditionHandle is invalid/unkwnown
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get condition state of the requested condition.	

⌋

8.41.2.1.3.2 SetCondition

[SWS_DM_01729] Definition of API function ara::diag::MultipleCondition::SetCondition

Upstream requirements: RS_AP_00139, RS_Diag_04192

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_condition.h"	
Scope:	`class ara::diag::MultipleCondition`	
Syntax:	`ara::core::Result< void > SetCondition (ConditionHandleType conditionHandle, ConditionType condition) noexcept;`	
Parameters (in):	conditionHandle	Identifies the requested condition
	condition	current condition
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided conditionHandle is invalid/unkwnown
Description:	Set condition state of the requested condition.	

⌋

8.41.2.1.3.3 operator=

[SWS_DM_02073] Definition of API function ara::diag::MultipleCondition::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_condition.h"
Scope:	`class ara::diag::MultipleCondition`
Syntax:	`MultipleCondition & operator= (MultipleCondition &)=delete;`
Description:	Copy assignment operator of MultipleCondition.

⌋

8.42 Header: ara/diag/multiple_event.h

8.42.1 Non-Member Types

8.42.1.1 Type Alias: EventHandleType

[SWS_DM_01703] Definition of API type ara::diag::EventHandleType

Upstream requirements: RS_Diag_04151

⌈

Kind:	type alias
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`namespace ara::diag`
Symbol:	EventHandleType
Syntax:	`using EventHandleType = std::uint32_t;`
Description:	Type to identify an event .

⌋

8.42.2 Class: MultipleEvent

[SWS_DM_01704] Definition of API class ara::diag::MultipleEvent

Upstream requirements: RS_Diag_04151

⌈

Kind:	class
Header file:	#include "ara/diag/multiple_event.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	MultipleEvent
Syntax:	`class MultipleEvent final {...};`
Description:	Class to implement operations on diagnostic Events with the multiple event interfaces.

⌋

8.42.2.1 Public Member Functions

8.42.2.1.1 Special Member Functions

8.42.2.1.1.1 Move Constructor

[SWS_DM_02069] Definition of API function ara::diag::MultipleEvent::MultipleEvent

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`
Syntax:	`MultipleEvent (MultipleEvent &&) noexcept=delete;`
Description:	Move constructor of MultipleEvent.

⌋

8.42.2.1.1.2 Move Assignment Operator

[SWS_DM_02068] Definition of API function ara::diag::MultipleEvent::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`
Syntax:	`MultipleEvent & operator= (MultipleEvent &&)=delete;`
Description:	Move assignment operator of MultipleEvent.

⌋

8.42.2.1.1.3 Destructor

[SWS_DM_01706] Definition of API function ara::diag::MultipleEvent::~MultipleEvent

Upstream requirements: RS_Diag_04151, RS_AP_00134

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`
Syntax:	`~MultipleEvent () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class MultipleEvent .

⌋

8.42.2.1.2 Constructors

8.42.2.1.2.1 MultipleEvent

[SWS_DM_02070] Definition of API function ara::diag::MultipleEvent::MultipleEvent

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`
Syntax:	`MultipleEvent (MultipleEvent &)=delete;`
Description:	Copy constructor of MultipleEvent.

⌋

8.42.2.1.2.2 MultipleEvent

[SWS_DM_01705] Definition of API function ara::diag::MultipleEvent::MultipleEvent

Upstream requirements: RS_Diag_04151, RS_AP_00137

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_event.h"	
Scope:	`class ara::diag::MultipleEvent`	
Syntax:	`explicit MultipleEvent (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMultipleEventPortMapping` needs to be typed by a `DiagnosticMultipleEventInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor fct. for objects of class MultipleEvent.	

⌋

8.42.2.1.3 Member Functions

8.42.2.1.3.1 GetDTCNumber

[SWS_DM_01709] Definition of API function ara::diag::MultipleEvent::GetDTCNumber

Upstream requirements: RS_Diag_04201, RS_AP_00139, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_event.h"	
Scope:	`class ara::diag::MultipleEvent`	
Syntax:	`ara::core::Future< std::uint32_t > GetDTCNumber (EventHandleType eventHandle, DTCFormatType dtcFormat) noexcept;`	
Parameters (in):	eventHandle	Identifies the requested event
	dtcFormat	Define DTC format for the return value.
Return value:	ara::core::Future< std::uint32_t >	DTC number in respective DTCFormatType
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided eventHandle is invalid/unkwnown
	DiagErrc::kNoSuchDTC	rollback_semantics
		No DTC available.
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the DTC-ID related to requested event.	

⌋

8.42.2.1.3.2 GetDebouncingStatus

[SWS_DM_01710] Definition of API function ara::diag::MultipleEvent::GetDebouncingStatus

Upstream requirements: RS_Diag_04068, RS_Diag_04225, RS_AP_00139

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`

▽

△

Syntax:	`ara::core::Future< DebouncingState > GetDebouncingStatus (EventHandleType eventHandle) noexcept;`	
Parameters (in):	eventHandle	Identifies the requested event
Return value:	ara::core::Future< DebouncingState >	Return the current debouncing state of this event.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided eventHandle is invalid/unkwnown
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get the current debouncing status of the requested event.	

⌋

8.42.2.1.3.3 GetEventStatus

[SWS_DM_01707] Definition of API function ara::diag::MultipleEvent::GetEvent Status

Upstream requirements: RS_Diag_04151, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_event.h"	
Scope:	`class ara::diag::MultipleEvent`	
Syntax:	`ara::core::Future< EventStatusByte > GetEventStatus (EventHandleType eventHandle) noexcept;`	
DIRECTION NOT DEFINED	eventHandle	--
Return value:	ara::core::Future< Event StatusByte >	the current diagnostic event status
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided eventHandle is invalid/unkwnown
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current status of the requested event.	

⌋

8.42.2.1.3.4 GetFaultDetectionCounter

[SWS_DM_01711] Definition of API function ara::diag::MultipleEvent::GetFaultDetectionCounter

Upstream requirements: RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_event.h"	
Scope:	`class ara::diag::MultipleEvent`	
Syntax:	`ara::core::Future< std::int8_t > GetFaultDetectionCounter (EventHandleType eventHandle) noexcept;`	
Parameters (in):	eventHandle	Identifies the requested event
Return value:	ara::core::Future< std::int8_t >	current FaultDetectionCounter value.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided eventHandle is invalid/unkwnown
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current value of Fault Detection Counter of the selected event.	

⌋

8.42.2.1.3.5 SetEventStatusChangedNotifier

[SWS_DM_01708] Definition of API function ara::diag::MultipleEvent::SetEventStatusChangedNotifier

Status: DRAFT
Upstream requirements: RS_Diag_04183, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_event.h"	
Scope:	`class ara::diag::MultipleEvent`	
Syntax:	`ara::core::Result< void > SetEventStatusChangedNotifier (EventHandleType eventHandle, std::function< void(EventStatusByte)> notifier) noexcept;`	
Parameters (in):	eventHandle	Identifies the requested event
	notifier	The function to be called if a diagnostic event is changed.

▽

△

Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		given argument is invalid (pointer).
	DiagErrc::kInvalid Argument	rollback_semantics
		The provided eventHandle is invalid/unkwnown
Description:	Register a notifier function which is called if a diagnostic event is changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.42.2.1.3.6 operator=

[SWS_DM_02067] Definition of API function ara::diag::MultipleEvent::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_event.h"
Scope:	`class ara::diag::MultipleEvent`
Syntax:	`MultipleEvent & operator= (MultipleEvent &)=delete;`
Description:	Copy assignment operator of MultipleEvent.

⌋

8.43 Header: ara/diag/multiple_monitor.h

8.43.1 Non-Member Types

8.43.1.1 Type Alias: MonitorHandleType

[SWS_DM_01701] Definition of API type ara::diag::MonitorHandleType

Upstream requirements: RS_Diag_04063

⌈

Kind:	type alias
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`namespace ara::diag`
Symbol:	MonitorHandleType
Syntax:	`using MonitorHandleType = std::uint32_t;`
Description:	Type to identify an monitor .

⌋

8.43.2 Class: MultipleMonitor

[SWS_DM_01694] Definition of API class ara::diag::MultipleMonitor

Upstream requirements: RS_Diag_04179

⌈

Kind:	class
Header file:	#include "ara/diag/multiple_monitor.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	MultipleMonitor
Syntax:	`class MultipleMonitor final {...};`
Description:	Class to implement monitor operations on the multiple monitor interfaces.

⌋

8.43.2.1 Public Member Functions

8.43.2.1.1 Special Member Functions

8.43.2.1.1.1 Move Constructor

[SWS_DM_02066] Definition of API function ara::diag::MultipleMonitor::MultipleMonitor

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`class ara::diag::MultipleMonitor`
Syntax:	`MultipleMonitor (MultipleMonitor &&) noexcept=delete;`
Description:	Move constructor of MultiMonitor.

⌋

8.43.2.1.1.2 Move Assignment Operator

[SWS_DM_02064] Definition of API function ara::diag::MultipleMonitor::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`class ara::diag::MultipleMonitor`
Syntax:	`MultipleMonitor & operator= (MultipleMonitor &&)=delete;`
Description:	Move assignment operator of MultiMonitor.

⌋

8.43.2.1.2 Constructors

8.43.2.1.2.1 MultipleMonitor

[SWS_DM_01695] Definition of API function ara::diag::MultipleMonitor::MultipleMonitor

Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`MultipleMonitor (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMultipleMonitorInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMultipleEventPortMapping` needs to be typed by a `DiagnosticMultipleMonitorInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	MultipleMonitor constructor allows to report status of more than one monitor. using ara::diag::CounterBased; using ara::diag::TimeBased;	

⌋

8.43.2.1.2.2 MultipleMonitor

[SWS_DM_02065] Definition of API function ara::diag::MultipleMonitor::MultipleMonitor

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`class ara::diag::MultipleMonitor`
Syntax:	`MultipleMonitor (MultipleMonitor &)=delete;`
Description:	Copy constructor of MultiMonitor.

⌋

8.43.2.1.3 Member Functions

8.43.2.1.3.1 ConfigureMonitor

[SWS_DM_01702] Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_Diag_04068, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`ara::core::Result< void > ConfigureMonitor (MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues) noexcept;`	
Parameters (in):	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
Return value:	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		The provided monitorHandle is invalid/unkwnown
Description:	Configures a monitor with counter-based debouncing. Any overloaded ConfigureMonitor shall be only called once per monitor. The monitor can only be used after this initialisation step.	

⌋

8.43.2.1.3.2 ConfigureMonitor

[SWS_DM_01697] Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_Diag_04225, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`ara::core::Result< void > ConfigureMonitor (MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, TimeBasedDefaultValues) noexcept;`	
Parameters (in):	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
Return value:	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		The provided monitorHandle is invalid/unkwnown
Description:	Configures a monitor with time-based debouncing. Any overloaded ConfigureMonitor shall be only called once per monitor. The monitor can only be used after this initialisation step.	

⌋

8.43.2.1.3.3 ConfigureMonitor

[SWS_DM_01973] Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`ara::core::Result< void > ConfigureMonitor (MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor) noexcept;`	
DIRECTION NOT DEFINED	monitorHandle	--

▽

△

	initMonitor	--
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Configures a monitor without debouncing. Any overloaded ConfigureMonitor shall be only called once per monitor. The monitor can only be used after this initialisation step. .	

⌋

8.43.2.1.3.4 ConfigureMonitor

[SWS_DM_01696] Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor

Status: DRAFT
Upstream requirements: RS_AP_00137, RS_Diag_04179, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`ara::core::Result< void > ConfigureMonitor (MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter) noexcept;`	
Parameters (in):	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetection Counter	Possibility to register a function to get the current FDC for this event.
Return value:	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		The provided monitorHandle is invalid/unkwnown
Description:	Configures a monitor with Monitor-internal debouncing. Any overloaded ConfigureMonitor shall be only called once per monitor. The monitor can only be used after this initialisation step.	

⌋

8.43.2.1.3.5 Offer

[SWS_DM_01699] Definition of API function ara::diag::MultipleMonitor::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.43.2.1.3.6 ReportMonitorAction

[SWS_DM_01698] Definition of API function ara::diag::MultipleMonitor::Report MonitorAction

Upstream requirements: RS_Diag_04179, RS_AP_00139, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	`class ara::diag::MultipleMonitor`	
Syntax:	`void ReportMonitorAction (MonitorHandleType monitorHandle, MonitorAction action) noexcept;`	
Parameters (in):	monitorHandle	Identifies the requested monitor
	action	Contains either the last (un-)qualified test result of the diagnostic monitor or commands to control the debouncing or to force a prestorage.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Function to report the status information being relevant for error monitoring paths. Calls with invalid monitorHandle are ignored.	

⌋

8.43.2.1.3.7 StopOffer

[SWS_DM_01700] Definition of API function ara::diag::MultipleMonitor::StopOffer

Upstream requirements: RS_Diag_04169

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`class ara::diag::MultipleMonitor`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.43.2.1.3.8 operator=

[SWS_DM_02063] Definition of API function ara::diag::MultipleMonitor::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/multiple_monitor.h"
Scope:	`class ara::diag::MultipleMonitor`
Syntax:	`MultipleMonitor & operator= (MultipleMonitor &)=delete;`
Description:	Copy assignment operator of MultiMonitor.

⌋

8.44 Header: ara/diag/indicator.h

8.44.1 Non-Member Types

8.44.1.1 Enumeration: IndicatorStatusType

[SWS_DM_00740] Definition of API enum ara::diag::IndicatorStatusType

Upstream requirements: RS_Diag_04204

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/indicator.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	IndicatorStatusType	
Underlying type:	--	
Syntax:	`enum class IndicatorStatusType {...};`	
Values:	kOff= 0x00	Indicator off mode {default}.
	kOnDemand= 0x06	Indicator on-demand mode.
Description:	Represents the state of an indicator.	

⌋

8.44.2 Class: Indicator

[SWS_DM_00741] Definition of API class ara::diag::Indicator

Upstream requirements: RS_Diag_04204

⌈

Kind:	class
Header file:	#include "ara/diag/indicator.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Indicator
Syntax:	`class Indicator final {...};`
Description:	DiagnosticIndicatorInterface provides functionality for handling indicators.

⌋

8.44.2.1 Public Member Functions

8.44.2.1.1 Special Member Functions

8.44.2.1.1.1 Move Constructor

[SWS_DM_01630] Definition of API function ara::diag::Indicator::Indicator

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/indicator.h"
Scope:	`class ara::diag::Indicator`
Syntax:	`Indicator (Indicator &&) noexcept=delete;`
Description:	Move constructor of Indicator.

⌋

8.44.2.1.1.2 Move Assignment Operator

[SWS_DM_01628] Definition of API function ara::diag::Indicator::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/indicator.h"
Scope:	`class ara::diag::Indicator`
Syntax:	`Indicator & operator= (Indicator &&)=delete;`
Description:	Move assignment operator of Indicator.

⌋

8.44.2.1.1.3 Destructor

[SWS_DM_00743] Definition of API function ara::diag::Indicator::~Indicator

Upstream requirements: RS_AP_00134, RS_Diag_04204

⌈

Kind:	function
Header file:	#include "ara/diag/indicator.h"
Scope:	`class ara::diag::Indicator`
Syntax:	`~Indicator () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticIndicatorInterface .

⌋

8.44.2.1.2 Constructors

8.44.2.1.2.1 Indicator

[SWS_DM_00742] Definition of API function ara::diag::Indicator::Indicator

Upstream requirements: RS_AP_00137, RS_Diag_04204

⌈

Kind:	function	
Header file:	#include "ara/diag/indicator.h"	
Scope:	`class ara::diag::Indicator`	
Syntax:	`explicit Indicator (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticIndicator Interface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticIndicatorPortMapping` needs to be typed by a `DiagnosticIndicatorInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"

▽

△

	InstanceSpecifier-AlreadyInUseViola-tion	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for DiagnosticIndicatorInterface.	

⌋

8.44.2.1.2.2 Indicator

[SWS_DM_01629] Definition of API function ara::diag::Indicator::Indicator

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/indicator.h"
Scope:	class ara::diag::Indicator
Syntax:	Indicator (Indicator &)=delete;
Description:	Indicator shall be a single not copy-able instance.

⌋

8.44.2.1.3 Member Functions

8.44.2.1.3.1 GetIndicatorState

[SWS_DM_00744] Definition of API function ara::diag::Indicator::GetIndicator State

Upstream requirements: RS_AP_00139, RS_Diag_04204

⌈

Kind:	function	
Header file:	#include "ara/diag/indicator.h"	
Scope:	class ara::diag::Indicator	
Syntax:	ara::core::Future< IndicatorStatusType > GetIndicatorState () noexcept;	
Return value:	ara::core::Future< IndicatorStatusType >	the current Indicator
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get current Indicator.	

⌋

8.44.2.1.3.2 SetNotifier

[SWS_DM_00745] Definition of API function ara::diag::Indicator::SetNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04204

⌈

Kind:	function	
Header file:	#include "ara/diag/indicator.h"	
Scope:	`class ara::diag::Indicator`	
Syntax:	`ara::core::Result< void > SetNotifier (std::function< void(IndicatorStatusType)> notifier) noexcept;`	
Parameters (in):	notifier	notifier function
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Register a notifier function which is called if the indicator is updated. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.44.2.1.3.3 operator=

[SWS_DM_01627] Definition of API function ara::diag::Indicator::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/indicator.h"
Scope:	`class ara::diag::Indicator`
Syntax:	`Indicator & operator= (Indicator &)=delete;`

▽

△

Description:	Indicator shall be a single not assignable instance.

⌋

8.45 Header: ara/diag/event.h

8.45.1 Class: Event

[SWS_DM_00646] Definition of API class ara::diag::Event

Upstream requirements: RS_Diag_04151

⌈

Kind:	class
Header file:	#include "ara/diag/event.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	Event
Syntax:	`class Event final {...};`
Description:	Class to implement operations on diagnostic Events.

⌋

8.45.1.1 Public Member Functions

8.45.1.1.1 Special Member Functions

8.45.1.1.1.1 Move Constructor

[SWS_DM_01682] Definition of API function ara::diag::Event::Event

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	`class ara::diag::Event`
Syntax:	`Event (Event &&) noexcept=delete;`
Description:	Move constructor of Event.

⌋

8.45.1.1.1.2 Move Assignment Operator

[SWS_DM_01680] Definition of API function ara::diag::Event::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	`class ara::diag::Event`
Syntax:	`Event & operator= (Event &&)=delete;`
Description:	Move assignment operator of Event.

⌋

8.45.1.1.1.3 Destructor

[SWS_DM_00648] Definition of API function ara::diag::Event::~Event

Upstream requirements: RS_Diag_04151, RS_AP_00134

⌈

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	`class ara::diag::Event`
Syntax:	`~Event () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class Event .

⌋

8.45.1.1.2 Constructors

8.45.1.1.2.1 Event

[SWS_DM_00647] Definition of API function ara::diag::Event::Event

Upstream requirements: RS_Diag_04151, RS_AP_00137

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`explicit Event (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticEventPortMapping` needs to be typed by a `DiagnosticEventInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor fct. for objects of class Event.	

⌋

8.45.1.1.2.2 Event

[SWS_DM_01681] Definition of API function ara::diag::Event::Event

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	`class ara::diag::Event`
Syntax:	`Event (Event &)=delete;`
Description:	Event shall be a single not copy-able instance.

⌋

8.45.1.1.3 Member Functions

8.45.1.1.3.1 GetDTCNumber

[SWS_DM_00653] Definition of API function ara::diag::Event::GetDTCNumber

Upstream requirements: RS_Diag_04201, RS_AP_00139, RS_AP_00119

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< std::uint32_t > GetDTCNumber (DTCFormatType dtcFormat) noexcept;`	
Parameters (in):	dtcFormat	Define DTC format for the return value.
Return value:	ara::core::Future< std::uint32_t >	DTC number in respective DTCFormatType
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kNoSuchDTC	rollback_semantics
		No DTC available.
	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the DTC-ID related to this event instance.	

⌋

8.45.1.1.3.2 GetDebouncingStatus

[SWS_DM_00654] Definition of API function ara::diag::Event::GetDebouncing Status

Upstream requirements: RS_Diag_04068, RS_Diag_04225, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< DebouncingState > GetDebouncingStatus () noexcept;`	
Return value:	ara::core::Future< DebouncingState >	Return the current debouncing state of this event.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get the current debouncing status .	

⌋

8.45.1.1.3.3 GetEventStatus

[SWS_DM_00649] Definition of API function ara::diag::Event::GetEventStatus

Upstream requirements: RS_Diag_04151, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< EventStatusByte > GetEventStatus () noexcept;`	
Return value:	ara::core::Future< EventStatusByte >	the current diagnostic event status
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current diagnostic event status.	

⌋

8.45.1.1.3.4 GetFaultDetectionCounter

[SWS_DM_00656] Definition of API function ara::diag::Event::GetFaultDetection Counter

Upstream requirements: RS_Diag_04068, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< std::int8_t > GetFaultDetectionCounter () noexcept;`	
Return value:	ara::core::Future< std::int8_t >	current FaultDetectionCounter value.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current value of Fault Detection Counter of this event.	

⌋

8.45.1.1.3.5 GetLatchedWIRStatus

[SWS_DM_00651] Definition of API function ara::diag::Event::GetLatched WIRStatus

Upstream requirements: RS_Diag_04204, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< bool > GetLatchedWIRStatus () noexcept;`	
Return value:	ara::core::Future< bool >	the current warning indicator status
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current warning indicator status.	

⌋

8.45.1.1.3.6 GetTestComplete

[SWS_DM_00655] Definition of API function ara::diag::Event::GetTestComplete

Upstream requirements: RS_Diag_04151, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Future< bool > GetTestComplete () noexcept;`	
Return value:	ara::core::Future< bool >	Return the current test_completed-state of this event. "true", if FDC = -128 or FDC = 127; "false" in all other cases.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get the status if the event has matured to test completed (corresponds to FDC = -128 or FDC = 127).	

⌋

8.45.1.1.3.7 SetEventStatusChangedNotifier

[SWS_DM_00650] Definition of API function ara::diag::Event::SetEventStatusChangedNotifier

Status: DRAFT

Upstream requirements: RS_Diag_04183, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Result< void > SetEventStatusChangedNotifier (std::function< void(EventStatusByte)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if a diagnostic event is changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		given argument is invalid (pointer).

▽

△

Description:	Register a notifier function which is called if a diagnostic event is changed. A consecutive call of this method will overwrite the previous registered notifier.

⌋

8.45.1.1.3.8 SetLatchedWIRStatus

[SWS_DM_00652] Definition of API function ara::diag::Event::SetLatched WIRStatus

Upstream requirements: RS_Diag_04151, RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	`class ara::diag::Event`	
Syntax:	`ara::core::Result< void > SetLatchedWIRStatus (bool status) noexcept;`	
Parameters (in):	status	Limp-home status as determined by the AA. '0' means limp-home not active; '1' means limp-home active;
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Set the warning indicator status.	

⌋

8.45.1.1.3.9 operator=

[SWS_DM_01679] Definition of API function ara::diag::Event::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	`class ara::diag::Event`
Syntax:	`Event & operator= (Event &)=delete;`
Description:	Event shall be a single not assignable instance.

⌋

8.46 Header: ara/diag/event_types.h

8.46.1 Non-Member Types

8.46.1.1 Enumeration: DTCFormatType

[SWS_DM_00642] Definition of API enum ara::diag::DTCFormatType

Upstream requirements: RS_Diag_04201, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/event_types.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DTCFormatType	
Underlying type:	std::uint8_t	
Syntax:	`enum class DTCFormatType : std::uint8_t {...};`	
Values:	kDTCFormatOBD= 0	SAE_J2012-DA_DTCFormat_00 as defined in ISO 15031-6 specification.
	kDTCFormatUDS= 1	ISO_14229-1_DTCFormat as defined in ISO 14229-1 specification.
	kDTCFormatJ1939= 2	SAE_J1939-73_DTCFormat as defined in SAE J1939-73.
Description:	Represents the type of the DTC format according to ISO 14229-1.	

⌋

8.46.1.2 Enumeration: DebouncingState

[SWS_DM_00645] Definition of API enum ara::diag::DebouncingState

Upstream requirements: RS_Diag_04068, RS_Diag_04225, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/event_types.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DebouncingState	
Underlying type:	std::uint8_t	
Syntax:	`enum class DebouncingState : std::uint8_t {...};`	
Values:	kNeutral= 0x00	Neutral (corresponds to FDC = 0)
	kTemporarilyDefective= 0x01	Temporarily Defective (corresponds to 0 < FDC < 127)
	kFinallyDefective= 0x02	finally Defective (corresponds to FDC = 127)

▽

△

	kTemporarilyHealed= 0x04	temporarily healed (corresponds to -128 < FDC < 0)
	kFinallyHealed= 0x08	finally healed (corresponds to FDC = -128)
Description:	Debounce status of event .	

⌋

8.46.1.3 Enumeration: EventStatusBit

[SWS_DM_00643] Definition of API enum ara::diag::EventStatusBit

Upstream requirements: RS_Diag_04151, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/event_types.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	EventStatusBit	
Underlying type:	std::uint8_t	
Syntax:	`enum class EventStatusBit : std::uint8_t {...};`	
Values:	kTestFailed	bit 0: TestFailed
	kTestFailedThis OperationCycle	bit 1: TestFailedThisOperationCycle
	kTestNotCompletedThis OperationCycle	bit 6: TestNotCompletedThisOperationCycle
Description:	Single event status bits.	

⌋

8.46.2 Struct: EventStatusByte

[SWS_DM_00644] Definition of API class ara::diag::EventStatusByte

Upstream requirements: RS_Diag_04151

⌈

Kind:	struct
Header file:	#include "ara/diag/event_types.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`

▽

△

Symbol:	EventStatusByte
Syntax:	`struct EventStatusByte {...};`
Description:	Current event status byte, bit-encoded.

⌋

8.46.2.1 Public Member Functions

8.46.2.1.1 Special Member Functions

8.46.2.1.1.1 Copy Constructor

[SWS_DM_01752] Definition of API function ara::diag::EventStatusByte::EventStatusByte

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr EventStatusByte (const EventStatusByte &) noexcept=default;`	
DIRECTION NOT DEFINED	const EventStatusByte &	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor .	

⌋

8.46.2.1.1.2 Move Constructor

[SWS_DM_01751] Definition of API function ara::diag::EventStatusByte::EventStatusByte

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr EventStatusByte (EventStatusByte &&) noexcept;`	
DIRECTION NOT DEFINED	EventStatusByte &&	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor .	

⌋

8.46.2.1.1.3 Copy Assignment Operator

[SWS_DM_01750] Definition of API function ara::diag::EventStatusByte::operator=

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr EventStatusByte & operator= (const EventStatusByte &) noexcept;`	
DIRECTION NOT DEFINED	const EventStatusByte &	--
Return value:	EventStatusByte &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator .	

⌋

8.46.2.1.1.4 Move Assignment Operator

[SWS_DM_01749] Definition of API function ara::diag::EventStatusByte::operator=

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr EventStatusByte & operator= (EventStatusByte &&) noexcept;`	
DIRECTION NOT DEFINED	EventStatusByte &&	--
Return value:	EventStatusByte &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator .	

⌋

8.46.2.1.2 Constructors

8.46.2.1.2.1 EventStatusByte

[SWS_DM_01753] Definition of API function ara::diag::EventStatusByte::EventStatusByte

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`template <typename... Args>` `constexpr EventStatusByte (Args... bits) noexcept;`	
DIRECTION NOT DEFINED	bits	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct an EventStatusByte where all EventStatusBits passed as parameter are set. .	

⌋

8.46.2.1.3 Member Functions

8.46.2.1.3.1 IsFailedAndTested

[SWS_DM_01757] Definition of API function ara::diag::EventStatusByte::IsFailedAndTested

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr bool IsFailedAndTested () const noexcept;`	
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether test is completed this operation cycle and failed. .	

⌋

8.46.2.1.3.2 IsNotSet

[SWS_DM_01754] Definition of API function ara::diag::EventStatusByte::IsNotSet

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`template <typename... Args>` `constexpr bool IsNotSet (Args... bits) const noexcept;`	
DIRECTION NOT DEFINED	bits	--
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether all EventStatusBits passed as parameter are unset. .	

⌋

8.46.2.1.3.3 IsPassedAndTested

[SWS_DM_01756] Definition of API function ara::diag::EventStatusByte::IsPassedAndTested

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`constexpr bool IsPassedAndTested () const noexcept;`	
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether test is completed this operation cycle and passed. .	

⌋

8.46.2.1.3.4 IsSet

[SWS_DM_01755] Definition of API function ara::diag::EventStatusByte::IsSet

Upstream requirements: RS_Diag_04151

⌈

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	`struct ara::diag::EventStatusByte`	
Syntax:	`template <typename... Args>` `constexpr bool IsSet (Args... bits) const noexcept;`	
DIRECTION NOT DEFINED	bits	--
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether all EventStatusBits passed as parameter are set. .	

⌋

8.47 Header: ara/diag/dtc_information.h

8.47.1 Non-Member Types

8.47.1.1 Enumeration: ControlDtcStatusType

[SWS_DM_00663] Definition of API enum ara::diag::ControlDtcStatusType

Upstream requirements: RS_Diag_04159

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/dtc_information.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	ControlDtcStatusType	
Underlying type:	std::uint8_t	
Syntax:	`enum class ControlDtcStatusType : std::uint8_t {...};`	
Values:	kDTCSettingOn= 0x00	Updating of diagnostic trouble code status bits is under normal operating conditions
	kDTCSettingOff= 0x01	Updating of diagnostic trouble code status bits is stopped.
Description:	Type for ControlDTCStatus status as requested by UDS service 0x85 ControlDTCSetting.	

⌋

8.47.1.2 Enumeration: UdsDtcStatusBitType

[SWS_DM_00658] Definition of API enum ara::diag::UdsDtcStatusBitType

Upstream requirements: RS_Diag_04067, RS_Diag_04151

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/dtc_information.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	UdsDtcStatusBitType	
Underlying type:	std::uint8_t	
Syntax:	`enum class UdsDtcStatusBitType : std::uint8_t {...};`	
Values:	kTestFailed= 0x01	bit 0: TestFailed
	kTestFailedThis OperationCycle= 0x02	bit 1: TestFailedThisOperationCycle
	kPendingDTC= 0x04	bit 2: PendingDTC
	kConfirmedDTC= 0x08	bit 3: ConfirmedDTC

▽

△

	kTestNotCompleted SinceLastClear= 0x10	bit 4: TestNotCompletedSinceLastClear
	kTestFailedSinceLast Clear= 0x20	bit 5: TestFailedSinceLastClear
	kTestNotCompletedThis OperationCycle= 0x40	bit 6: TestNotCompletedThisOperationCycle
	kWarningIndicator Requested= 0x80	bit 7: WarningIndicatorRequested
Description:	UDS DTC status bits according to ISO 14229-1.	

⌋

8.47.2 Class: DTCInformation

[SWS_DM_00657] Definition of API class ara::diag::DTCInformation

Upstream requirements: RS_Diag_04150, RS_Diag_04105

⌈

Kind:	class
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DTCInformation
Syntax:	`class DTCInformation final {...};`
Description:	Class to implement operations on DTC informations per configured DiagnosticMemory Destination.

⌋

8.47.2.1 Public Member Types

8.47.2.1.1 Type Alias: EventMemoryOverflowSetNotifier

[SWS_DM_02076] Definition of API type ara::diag::DTCInformation::EventMemoryOverflowSetNotifier

Upstream requirements: RS_AP_00139, RS_Diag_04093

⌈

Kind:	type alias
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Symbol:	EventMemoryOverflowSetNotifier
Syntax:	`using EventMemoryOverflowSetNotifier = std::function<void(bool)>;`
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the current event memory overflow status changed. .

⌋

8.47.2.2 Public Member Functions

8.47.2.2.1 Special Member Functions

8.47.2.2.1.1 Move Constructor

[SWS_DM_01614] Definition of API function ara::diag::DTCInformation::DTCInformation

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Syntax:	`DTCInformation (DTCInformation &&) noexcept=delete;`
Description:	Move constructor of DTCInformation.

⌋

8.47.2.2.1.2 Move Assignment Operator

[SWS_DM_01612] Definition of API function ara::diag::DTCInformation::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Syntax:	`DTCInformation & operator= (DTCInformation &&)=delete;`
Description:	Move assignment operator of DTCInformation.

⌋

8.47.2.2.1.3 Destructor

[SWS_DM_00665] Definition of API function ara::diag::DTCInformation::~DTCInformation

Upstream requirements: RS_AP_00134

⌈

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Syntax:	`~DTCInformation () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class DTCInformation .

⌋

8.47.2.2.2 Constructors

8.47.2.2.2.1 DTCInformation

[SWS_DM_00664] Definition of API function ara::diag::DTCInformation::DTCInformation

Upstream requirements: RS_AP_00137, RS_Diag_04150, RS_Diag_04105

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`explicit DTCInformation (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an Diagnostic DTCInformationInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticMemoryDestinationPortMapping` needs to be typed by a `DiagnosticDTCInformationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for a DTCInformation instance which allows for DTC related operation per DiagnosticMemoryDestination.	

⌋

8.47.2.2.2.2 DTCInformation

[SWS_DM_01613] Definition of API function ara::diag::DTCInformation::DTCInformation

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Syntax:	`DTCInformation (DTCInformation &)=delete;`
Description:	DTCInformation shall be a single not copy-able instance.

⌋

8.47.2.2.3 Member Functions

8.47.2.2.3.1 Clear

[SWS_DM_00671] Definition of API function ara::diag::DTCInformation::Clear

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04194

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Future< void > Clear (std::uint32_t dtcGroup) noexcept;`	
Parameters (in):	dtcGroup	DTC group to be cleared.
Return value:	ara::core::Future< void >	void or errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kBusy	rollback_semantics
		Busy processing.
	DiagErrc::kFailed	rollback_semantics
		Clear failed.
	DiagErrc::kMemoryError	rollback_semantics
		Memory error reported.
	DiagErrc::kWrongDtc	rollback_semantics
		Wrong DTC group passed.
	ara::diag::DiagErrc::k ServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Method for Clearing a DTC or a group of DTCs.	

⌋

8.47.2.2.3.2 EnableControlDtc

[SWS_DM_00674] Definition of API function ara::diag::DTCInformation::EnableControlDtc

Upstream requirements: RS_AP_00139, RS_Diag_04159

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > EnableControlDtc () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Triggers a setting of ControlDTCStatus to enabled in case the application has some conditions or states demands to do so. The control DTC setting state might be set to enabled only after this API has returned. .	

⌋

8.47.2.2.3.3 GetControlDTCStatus

[SWS_DM_00672] Definition of API function ara::diag::DTCInformation::GetControlDTCStatus

Upstream requirements: RS_AP_00139, RS_Diag_04159

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Future< ControlDtcStatusType > GetControlDTCStatus () noexcept;`	
Return value:	ara::core::Future< ControlDtcStatusType >	The current status of ControlDtcStatus (related to UDS service 0x85)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.
Description:	Contains the current status of the ControlDTCStatus.	

⌋

8.47.2.2.3.4 GetCurrentStatus

[SWS_DM_00666] Definition of API function ara::diag::DTCInformation::GetCurrentStatus

Upstream requirements: RS_AP_00139

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Future< UdsDtcStatusByteType > GetCurrentStatus (std::uint32_t dtc) noexcept;`	
Parameters (in):	dtc	DTC indentifier for which the status should be retrieved.
Return value:	ara::core::Future< UdsDtcStatusByteType >	the current UDS DTC status byte of the given DTC identifier.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kNoSuchDtc	rollback_semantics
		given DTC identifier not available.
	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Retrieves the current UDS DTC status byte of the given DTC identifier.	

⌋

8.47.2.2.3.5 GetEventMemoryOverflow

[SWS_DM_00919] Definition of API function ara::diag::DTCInformation::GetEventMemoryOverflow

Upstream requirements: RS_AP_00139, RS_Diag_04093

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< bool > GetEventMemoryOverflow () noexcept;`	
Return value:	ara::core::Result< bool >	Current status of event memory overflow.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics

▽

△

		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Contains the current event memory overflow status.	

⌋

8.47.2.2.3.6 GetNumberOfStoredEntries

[SWS_DM_00669] Definition of API function ara::diag::DTCInformation::GetNumberOfStoredEntries

Upstream requirements: RS_AP_00139, RS_Diag_04109

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Future< std::uint32_t > GetNumberOfStoredEntries () noexcept;`	
Return value:	ara::core::Future< std::uint32_t >	Number of currently stored fault memory entries.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ara::diag::DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Contains the number of currently stored fault memory entries.	

⌋

8.47.2.2.3.7 SetControlDtcStatusNotifier

[SWS_DM_00673] Definition of API function ara::diag::DTCInformation::SetControlDtcStatusNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04159

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > SetControlDtcStatusNotifier (std::function< void(ControlDtcStatusType)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if the ControlDTCStatus (related to UDS service 0x85) for this diagnostic memory instance has changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		given argument is invalid (pointer).
Description:	Registers a notifier function which is called if the control DTC setting is changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.47.2.2.3.8 SetDTCStatusChangedNotifier

[SWS_DM_00667] Definition of API function ara::diag::DTCInformation::SetDTCStatusChangedNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04148

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > SetDTCStatusChangedNotifier (std::function< void(std::uint32_t dtc, UdsDtcStatusByteType udsStatusByteOld, UdsDtcStatusByteType udsStatusByteNew)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if a DTC status has changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Register a notifier function which is called if a UDS DTC status is changed. A consecutive call of this method will overwrite the previous registered notifier.

⌋

8.47.2.2.3.9 SetEventMemoryOverflowNotifier

[SWS_DM_00918] Definition of API function ara::diag::DTCInformation::SetEventMemoryOverflowNotifier

Upstream requirements: RS_AP_00139, RS_Diag_04093

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > SetEventMemoryOverflowNotifier (EventMemoryOverflowSetNotifier notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if the overflow status for this diagnostic event memory instance has changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics given argument is invalid (pointer).
Description:	Register a notifier function which is called if the current event memory overflow status changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.47.2.2.3.10 SetNumberOfStoredEntriesNotifier

[SWS_DM_00670] Definition of API function ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04109

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > SetNumberOfStoredEntriesNotifier (std::function< void(std::uint32_t)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if the number of entries for this diagnostic event memory instance has changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics given argument is invalid (pointer).
Description:	Register a notifier function which is called if the number of currently stored fault memory entries changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.47.2.2.3.11 SetSnapshotRecordUpdatedNotifier

[SWS_DM_00668] Definition of API function ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier

Status: DRAFT
Upstream requirements: RS_AP_00139, RS_Diag_04205, RS_Diag_04091

⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`class ara::diag::DTCInformation`	
Syntax:	`ara::core::Result< void > SetSnapshotRecordUpdatedNotifier (std::function< void(SnapshotRecordUpdatedType)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if the SnapshotRecord is changed.
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics

▽

△

		given argument is invalid (pointer).
Description:	Register a notifier function which is called if the SnapshotRecord is changed. A consecutive call of this method will overwrite the previous registered notifier.	

⌋

8.47.2.2.3.12 operator=

[SWS_DM_01611] Definition of API function ara::diag::DTCInformation::operator=

Upstream requirements: RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	`class ara::diag::DTCInformation`
Syntax:	`DTCInformation & operator= (DTCInformation &)=delete;`
Description:	DTCInformation shall be a single not assignable instance.

⌋

8.47.3 Struct: SnapshotDataIdentifierType

[SWS_DM_00660] Definition of API class ara::diag::DTCInformation::SnapshotDataIdentifierType

Upstream requirements: RS_Diag_04205, RS_Diag_04091

⌈

Kind:	struct
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DTCInformation`
Symbol:	SnapshotDataIdentifierType
Syntax:	`struct SnapshotDataIdentifierType {...};`
Description:	Type for SnapshotDataIdentifierType status.

⌋

8.47.4 Struct: SnapshotDataRecordType

[SWS_DM_00661] Definition of API class ara::diag::DTCInformation::SnapshotDataRecordType

Upstream requirements: RS_Diag_04205, RS_Diag_04091

⌈

Kind:	struct
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DTCInformation`
Symbol:	SnapshotDataRecordType
Syntax:	`struct SnapshotDataRecordType {...};`
Description:	Type for SnapshotDataRecordType status.

⌋

8.47.5 Struct: SnapshotRecordUpdatedType

[SWS_DM_00662] Definition of API class ara::diag::DTCInformation::SnapshotRecordUpdatedType

Upstream requirements: RS_Diag_04205, RS_Diag_04091

⌈

Kind:	struct
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DTCInformation`
Symbol:	SnapshotRecordUpdatedType
Syntax:	`struct SnapshotRecordUpdatedType {...};`
Description:	Type for SnapshotRecordUpdatedType status.

⌋

8.47.6 Struct: UdsDtcStatusByteType

[SWS_DM_00659] Definition of API class ara::diag::DTCInformation::UdsDtcStatusByteType

Upstream requirements: RS_Diag_04067, RS_Diag_04151

⌈

Kind:	struct
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::DTCInformation`
Symbol:	UdsDtcStatusByteType
Syntax:	`struct UdsDtcStatusByteType {...};`
Description:	Type for UDS DTC status byte.

⌋

8.47.6.1 Public Member Functions

8.47.6.1.1 Special Member Functions

8.47.6.1.1.1 Move Constructor

[SWS_DM_02086] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr UdsDtcStatusByteType (UdsDtcStatusByteType &&) noexcept;`	
DIRECTION NOT DEFINED	UdsDtcStatusByteType &&	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor.	

⌋

8.47.6.1.1.2 Copy Constructor

[SWS_DM_02085] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr UdsDtcStatusByteType (const UdsDtcStatusByteType &) noexcept=default;`	
DIRECTION NOT DEFINED	const UdsDtcStatusByteType &	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor.	

⌋

8.47.6.1.1.3 Copy Assignment Operator

[SWS_DM_02087] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::operator= ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr UdsDtcStatusByteType & operator= (const UdsDtcStatusByteType &) noexcept;`	
DIRECTION NOT DEFINED	const UdsDtcStatusByteType &	--
Return value:	UdsDtcStatusByteType &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator .	

⌋

8.47.6.1.1.4 Move Assignment Operator

[SWS_DM_02088] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::operator= ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr UdsDtcStatusByteType & operator= (UdsDtcStatusByteType &&) noexcept;`	
DIRECTION NOT DEFINED	UdsDtcStatusByteType &&	--
Return value:	UdsDtcStatusByteType &	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator .	

⌋

8.47.6.1.2 Constructors

8.47.6.1.2.1 UdsDtcStatusByteType

[SWS_DM_02084] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`template <typename... Args>` `constexpr UdsDtcStatusByteType (Args... bits) noexcept;`	
DIRECTION NOT DEFINED	bits	--
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct an UdsDtcStatusByteType where all DtcStatusBits passed as parameter are set. .	

⌋

8.47.6.1.3 Member Functions

8.47.6.1.3.1 IsFailedAndTested

[SWS_DM_02080] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsFailedAndTested ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr bool IsFailedAndTested () const noexcept;`	
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Test if DTC is Failed and Tested.	

⌋

8.47.6.1.3.2 IsNotSet

[SWS_DM_02083] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsNotSet ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`template <typename... Args>` `constexpr bool IsNotSet (Args... bits) const noexcept;`	
DIRECTION NOT DEFINED	bits	--
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Test if DTC status bit(s) is not set.	

⌋

8.47.6.1.3.3 IsPassedAndTested

[SWS_DM_02081] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsPassedAndTested ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`constexpr bool IsPassedAndTested () const noexcept;`	
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Test if DTC is Passed and Tested.	

⌋

8.47.6.1.3.4 IsSet

[SWS_DM_02082] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsSet ⌈

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	`struct ara::diag::DTCInformation::UdsDtcStatusByteType`	
Syntax:	`template <typename... Args>` `constexpr bool IsSet (Args... bits) const noexcept;`	
DIRECTION NOT DEFINED	bits	--
Return value:	bool	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Test if DTC status bit(s) is set.	

⌋

8.48 Diagnostic SOVD-related APIs

8.49 Header: ara/diag/diag_sovd_error_domain.h

8.49.1 Non-Member Types

8.49.1.1 Enumeration: DiagSovdErrc

[SWS_DM_01806] Definition of API enum ara::diag::DiagSovdErrc

Upstream requirements: RS_AP_00119, RS_AP_00125

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	DiagSovdErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class DiagSovdErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kInvalidSignature= 1	The signature of the data in the payload is invalid.
	kSovdServer Misconfigured= 2	The SOVD server is not configured correctly, e.g. required configuration files or other data is missing.
	kTemporarilyNot Available= 3	The request currently can't be served.
	kInvalidAccessToken= 4	The access token provided by the client is invalid.
	kInsufficientAccess Rights= 5	The SOVD client does not have the right to access the resource.
	kPreconditionNot Fulfilled= 6	The preconditions to execute the method are not fulfilled.
	kUpdateProcessIn Progress= 7	An update is already in progress and not yet done or aborted.
	kUpdateAutomatedNot Supported= 8	Automatic installation of update is not supported.
	kUpdatePreparationIn Progress= 9	An update is already in preparation and not yet done or aborted.
	kUpdateExecutionIn Progress= 10	Another update is currently executed and not yet done or aborted.
Description:	Specifies the types of errors that can occur in SOVD services.	

⌋

8.49.2 Non-Member Functions

8.49.2.1 Other

8.49.2.1.1 GetDiagSovdErrorDomain

[SWS_DM_01816] Definition of API function ara::diag::GetDiagSovdErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetDiagSovdErrorDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	reference to the DiagSovdErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagSovdErrorDomain instance.	

⌋

8.49.2.1.2 MakeErrorCode

[SWS_DM_01817] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`namespace ara::diag`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (DiagSovdErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	an enumeration value from DiagSovdErrc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Create a new ErrorCode for DiagSovdErrorDomain with the given support data type and message.	

⌋

8.49.3 Class: DiagSovdErrorDomain

[SWS_DM_01809] Definition of API class ara::diag::DiagSovdErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagSovdErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class DiagSovdErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::diag::DiagSovdErrorDomain` in [SWS_CORE_90023]
Description:	Error domain for errors originating from SOVD.

⌋

8.49.3.1 Public Member Types

8.49.3.1.1 Type Alias: Errc

[SWS_DM_01810] Definition of API type ara::diag::DiagSovdErrorDomain::Errc

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	`class ara::diag::DiagSovdErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = DiagSovdErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.49.3.1.2 Type Alias: Exception

[SWS_DM_01811] Definition of API type ara::diag::DiagSovdErrorDomain::Exception

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	type alias
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	`class ara::diag::DiagSovdErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = DiagSovdException;`
Description:	Alias for the exception base class.

⌋

8.49.3.2 Public Member Functions

8.49.3.2.1 Special Member Functions

8.49.3.2.1.1 Default Constructor

[SWS_DM_01812] Definition of API function ara::diag::DiagSovdErrorDomain::DiagSovdErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	`class ara::diag::DiagSovdErrorDomain`
Syntax:	`constexpr DiagSovdErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

⌋

8.49.3.2.2 Member Functions

8.49.3.2.2.1 Message

[SWS_DM_01814] Definition of API function ara::diag::DiagSovdErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`class ara::diag::DiagSovdErrorDomain`	
Syntax:	`const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Translate an error code value into a text message.	

⌋

8.49.3.2.2.2 Name

[SWS_DM_01813] Definition of API function ara::diag::DiagSovdErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`class ara::diag::DiagSovdErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"DiagSovdErrorDomain"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

⌋

8.49.3.2.2.3 ThrowAsException

[SWS_DM_01815] Definition of API function ara::diag::DiagSovdErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`class ara::diag::DiagSovdErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given ErrorCode. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.49.4 Class: DiagSovdException

[SWS_DM_01807] Definition of API class ara::diag::DiagSovdException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	class
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	DiagSovdException
Base class:	ara::core::Exception
Syntax:	`class DiagSovdException : public ara::core::Exception {...};`
Description:	Exception type for SOVD thrown by Diag classes.

⌋

8.49.4.1 Public Member Functions

8.49.4.1.1 Constructors

8.49.4.1.1.1 DiagSovdException

[SWS_DM_01808] Definition of API function ara::diag::DiagSovdException::DiagSovdException

Upstream requirements: RS_AP_00119, RS_AP_00132

⌈

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	`class ara::diag::DiagSovdException`	
Syntax:	`explicit DiagSovdException (ara::core::ErrorCode err) noexcept;`	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagSovdException from an ErrorCode.	

⌋

8.50 Header: ara/diag/sovd_authorization.h

8.50.1 Class: SovdAuthorization

[SWS_DM_01490] Definition of API class ara::diag::SovdAuthorization

Upstream requirements: RS_Diag_04268

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_authorization.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdAuthorization
Syntax:	`class SovdAuthorization {...};`
Description:	Class to implement the SOVD Authorization interfaces in the application.

⌋

8.50.1.1 Public Member Functions

8.50.1.1.1 Special Member Functions

8.50.1.1.1.1 Destructor

[SWS_DM_01488] Definition of API function ara::diag::SovdAuthorization::~SovdAuthorization

Upstream requirements: RS_Diag_04268

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_authorization.h"
Scope:	`class ara::diag::SovdAuthorization`
Syntax:	`virtual ~SovdAuthorization () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdAuthorization .

⌋

8.50.1.1.2 Constructors

8.50.1.1.2.1 SovdAuthorization

[SWS_DM_01489] Definition of API function ara::diag::SovdAuthorization::SovdAuthorization

Upstream requirements: RS_Diag_04268

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	`class ara::diag::SovdAuthorization`	
Syntax:	`explicit SovdAuthorization (ara::core::InstanceSpecifier instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a `DiagnosticSovdAuthorizationInterface` service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticSovdAuthorizationPortMapping` needs to be typed by a `DiagnosticSovdAuthorizationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructs the port for SOVD Authorization interface.	

⌋

8.50.1.1.3 Member Functions

8.50.1.1.3.1 GetAuthorizationUrl

[SWS_DM_01485] Definition of API function ara::diag::SovdAuthorization::GetAuthorizationUrl

Upstream requirements: RS_Diag_04268

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	`class ara::diag::SovdAuthorization`	
Syntax:	`virtual ara::core::Future< ara::core::String > GetAuthorizationUrl (const ara::core::Map< const ara::core::StringView, const ara::core::Vector< const ara::core::StringView > > &queryParameters, const MetaInfo &metaInfo) noexcept=0;`	
Parameters (in):	queryParameters	The query parameters passed in the request.
	metaInfo	The meta information of a diagnostic service port.
Return value:	ara::core::Future< ara::core::String >	The URL that the SOVD client shall use for authorization.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor

▽

△

Description:	Get the URL that the SOVD client shall use for authorization. The SOVD client will be redirected to that URL. If OAuth 2.0 is used, this would be the URL to look up the Authorization Server's authorize endpoint.

⌋

8.50.1.1.3.2 GetTokenUrl

[SWS_DM_01484] Definition of API function ara::diag::SovdAuthorization::GetTokenUrl

Upstream requirements: RS_Diag_04268

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	`class ara::diag::SovdAuthorization`	
Syntax:	`virtual ara::core::Future< ara::core::String > GetTokenUrl (const ara::core::Map< const ara::core::StringView, const ara::core::Vector< const ara::core::StringView > > &queryParameters, const MetaInfo &metaInfo) noexcept=0;`	
Parameters (in):	queryParameters	The query parameters passed in the request.
	metaInfo	The meta information of a diagnostic service port.
Return value:	ara::core::Future< ara::core::String >	The URL that the SOVD client shall use to request a token.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Get the URL that the SOVD client shall use to request a token. The SOVD client will be redirected to that URL. If OAuth 2.0 is used, this would be the URL to look up the Authorization Server's token endpoint.	

⌋

8.50.1.1.3.3 Offer

[SWS_DM_01487] Definition of API function ara::diag::SovdAuthorization::Offer

Upstream requirements: RS_Diag_04268

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	`class ara::diag::SovdAuthorization`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics
		Implementation does not fit to the configuration.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.50.1.1.3.4 StopOffer

[SWS_DM_01486] Definition of API function ara::diag::SovdAuthorization::Stop Offer

Upstream requirements: RS_Diag_04268

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_authorization.h"
Scope:	`class ara::diag::SovdAuthorization`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.50.1.1.3.5 ValidateAuthorization

[SWS_DM_01483] Definition of API function ara::diag::SovdAuthorization::ValidateAuthorization

Upstream requirements: RS_Diag_04268

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	`class ara::diag::SovdAuthorization`	
Syntax:	`virtual ara::core::Future< ValidateAuthorizationOutput > ValidateAuthorization (ara::core::StringView token, const MetaInfo &metaInfo, ClientAuthentication &clientAuthentication) noexcept=0;`	
Parameters (in):	token	The authorization token sent by the SOVD client in the HTTP Authorization header.
	metaInfo	The meta information of a diagnostic service port.
	clientAuthentication	Object used by the application to report the authenticated state of the SOVD client. After the returned future is ready the referenced object shall not be used any more by the application.
Return value:	ara::core::Future< ValidateAuthorization Output >	Either ValidateAuthorizationOutput (for a successful validation) or an error code.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	ara::diag::DiagSovd Errc::kInvalid	rollback_semantics
		AccessToken, In case the access token is invalid.
	ara::diag::DiagSovd Errc::kTemporarilyNot Available	rollback_semantics
		In case the access token currently can't be verified (e.g. because online verification is needed and the authentication server can't be reached).
Description:	This function validates the authorization token sent by the client. The application shall validate the authorization token according to the authorization protocol used in the system and then according to the outcome set the respective roles as authorized in the clientAuthentication that is passed as parameter. If OAuth 2.0 is used, this would implement the validation that is supposed to happen in the Resource Server.	

⌋

8.50.2 Struct: ValidateAuthorizationOutput

[SWS_DM_01927] Definition of API class ara::diag::SovdAuthorization::ValidateAuthorizationOutput

Upstream requirements: RS_Diag_04268

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_authorization.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdAuthorization`
Symbol:	ValidateAuthorizationOutput
Syntax:	`struct ValidateAuthorizationOutput {...};`
Description:	Return type for ValidateAuthorization().

⌋

8.50.2.1 Public Member Variables

8.50.2.1.1 identity

[SWS_DM_01819] Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorizationOutput::identity

Upstream requirements: RS_Diag_04268

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_authorization.h"
Scope:	`struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput`
Symbol:	identity
Type:	`ara::core::Optional< ara::core::String >`
Syntax:	`ara::core::Optional<ara::core::String> identity;`
Description:	Identity of the client. This identity string if present will be used to re-identify a client even if it uses a new token (e.g. because the old one expired) so that it has access to the temporary resources (e.g. running routines, locks, etc.). If the identity string is the same as one returned earlier, the client will be considered identical. Depending on the security concept it could be for example the 'client_id', the 'sub', or a combination of both. If the identity string is not provided, the client will only have access to the temporary resources as long as it provides the same access token and validUntil is not yet expired.

⌋

8.50.2.1.2 validUntil

[SWS_DM_01820] Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorizationOutput::validUntil

Upstream requirements: RS_Diag_04268

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_authorization.h"
Scope:	`struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput`
Symbol:	validUntil
Type:	`std::chrono::system_clock::time_point`
Syntax:	`std::chrono::system_clock::time_point validUntil;`
Description:	Time until this validation may be reused. The point in time until which the authentication state may be reused if the same token is provided by the client (caching).

⌋

8.51 Header: ara/diag/sovd_bulk_data.h

8.51.1 Class: SovdBulkData

[SWS_DM_01841] Definition of API class ara::diag::SovdBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_bulk_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdBulkData
Syntax:	`class SovdBulkData {...};`
Description:	Generic SOVD Bulk Data interface.

⌋

8.51.1.1 Public Member Functions

8.51.1.1.1 Special Member Functions

8.51.1.1.1.1 Destructor

[SWS_DM_01852] Definition of API function ara::diag::SovdBulkData::~SovdBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`class ara::diag::SovdBulkData`
Syntax:	`virtual ~SovdBulkData () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdBulkData .

⌋

8.51.1.1.2 Constructors

8.51.1.1.2.1 SovdBulkData

[SWS_DM_01847] Definition of API function ara::diag::SovdBulkData::SovdBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`SovdBulkData (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a `DiagnosticSovdBulkDataInterface`
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticSovdBulkDataPortMapping` needs to be typed by a `DiagnosticSovdBulkDataInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for the SovdBulkData Interface.	

⌋

8.51.1.1.3 Member Functions

8.51.1.1.3.1 DeleteAllBulkData

[SWS_DM_01855] Definition of API function ara::diag::SovdBulkData::DeleteAllBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`virtual ara::core::Future< DeleteByCategoryResult > DeleteAllBulkData (ara::core::String bulkDataCategory, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	bulkDataCategory	Bulk data category of which all instances shall be deleted
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< DeleteByCategoryResult >	A future with a DeleteByCategoryResult or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Delete All Bulk Data Defined by Category".	

⌋

8.51.1.1.3.2 DeleteSpecificBulkData

[SWS_DM_01856] Definition of API function ara::diag::SovdBulkData::DeleteSpecificBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`virtual ara::core::Future< void > DeleteSpecificBulkData (ara::core::String bulkDataCategory, ara::core::String bulkDataId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	bulkDataCategory	Bulk data category of the bulk data instance to be be deleted
	bulkDataId	Bulk data bulk-data-id of the bulk data instance to be deleted
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< void >	A future with either a void data result for a successfull deletion or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Delete Specific Bulk Data Resource".	

⌋

8.51.1.1.3.3 GetBulkDataMetaData

[SWS_DM_01853] Definition of API function ara::diag::SovdBulkData::GetBulkDataMetaData

Status: DRAFT
Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`virtual ara::core::Future< ara::core::Span< BulkDataMetaDataDescriptor > > GetBulkDataMetaData (ara::core::String bulkDataCategory, MetaInfo const &metaInfo, ara::diag::ReleaseHandler releaseHandler, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	bulkDataCategory	Bulk data category to be read
	metaInfo	Contains additional meta information

▽

△

	releaseHandler	releaseHandler to notify when the result Span can be modified again
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< ara::core::Span< Bulk DataMetaDataDescriptor > >	A future with a span of all bulk data instances of the requested bulk data category. The span shall not be modified until the release Handler is called.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Read Bulk Data Meta Data".	

⌋

8.51.1.1.3.4 Offer

[SWS_DM_01857] Definition of API function ara::diag::SovdBulkData::Offer

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	Positive result if service was offered, error if offer failed
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics
		if handle is an invalid handle
	DiagOfferErrc::kAlready Offered	rollback_semantics
		if service has been offered already
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.51.1.1.3.5 RequestBulkDataDownload

[SWS_DM_01873] Definition of API function ara::diag::SovdBulkData::RequestBulkDataDownload

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`virtual ara::core::Future< ara::core::Variant< std::tuple< DataTransferReadSession, BulkDataMetaDataDescriptor >, HttpRedirect > > RequestBulkDataDownload (ara::core::String bulkDataCategory, ara::core::String bulkDataId, ara::core::Vector< ara::core::String > acceptHeader, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	bulkDataCategory	Bulk data category of the bulk data instance to be downloaded
	bulkDataId	Bulk data bulk-data-id of the bulk data instance to be downloaded
	acceptHeader	Lists the Accept request HTTP header
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< ara::core::Variant< std::tuple< DataTransferReadSession, BulkDataMetaDataDescriptor >, HttpRedirect > >	A future with either an instance of a "SOVD file reading session" with the corresponding BulkDataMetaDataDescriptor, an HTTP redirect or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Download Bulk Data".	

⌋

8.51.1.1.3.6 RequestBulkDataUpload

[SWS_DM_01854] Definition of API function ara::diag::SovdBulkData::RequestBulkDataUpload

Upstream requirements: RS_Diag_04266

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`class ara::diag::SovdBulkData`

▽

△

Syntax:	`virtual ara::core::Future< std::tuple< DataTransferWriteSession, BulkDataMetaDataDescriptor > > RequestBulkDataUpload (ara::core::String bulkDataCategory, BulkDataMetaDataDescriptor bulkDataMetaData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	bulkDataCategory	Bulk data category of which a new bulk data instance shall be uploaded
	bulkDataMetaData	BulkDataMetaDataDescriptor where the available information from the SOVD POST Request headers are prefilled
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< std::tuple< DataTransfer WriteSession, BulkData MetaDataDescriptor > >	A future with either an instance of a "SOVD file writing session" with the corresponding BulkDataMetaDataDescriptor or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Upload Bulk Data".	

⌋

8.51.1.1.3.7 StopOffer

[SWS_DM_01858] Definition of API function ara::diag::SovdBulkData::StopOffer

Upstream requirements: RS_Diag_04266

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`class ara::diag::SovdBulkData`
Syntax:	`void StopOffer () const noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

⌋

8.51.1.2 Protected Member Functions

8.51.1.2.1 Special Member Functions

8.51.1.2.1.1 Copy Constructor

[SWS_DM_01848] Definition of API function ara::diag::SovdBulkData::SovdBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`class ara::diag::SovdBulkData`
Syntax:	`SovdBulkData (SovdBulkData const &)=delete;`
Description:	SovdBulkData shall be a single not copy-able instance.
Visibility:	protected

⌋

8.51.1.2.1.2 Move Constructor

[SWS_DM_01849] Definition of API function ara::diag::SovdBulkData::SovdBulkData

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`SovdBulkData (SovdBulkData &&other) noexcept;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructs an instance of SovdBulkData.	
Visibility:	protected	

⌋

8.51.1.2.1.3 Copy Assignment Operator

[SWS_DM_01850] Definition of API function ara::diag::SovdBulkData::operator=

Upstream requirements: RS_Diag_04266

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`class ara::diag::SovdBulkData`
Syntax:	`SovdBulkData & operator= (SovdBulkData const &)=delete;`
Description:	SovdBulkData shall be a single not assignable instance.
Visibility:	protected

⌋

8.51.1.2.1.4 Move Assignment Operator

[SWS_DM_01851] Definition of API function ara::diag::SovdBulkData::operator=

Upstream requirements: RS_Diag_04266

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	`class ara::diag::SovdBulkData`	
Syntax:	`SovdBulkData & operator= (SovdBulkData &&other) & noexcept;`	
Parameters (out):	other	The other object
Return value:	SovdBulkData &	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of SovdBulkData.	
Visibility:	protected	

⌋

8.51.2 Struct: BulkDataMetaDataDescriptor

[SWS_DM_01830] Definition of API class ara::diag::SovdBulkData::BulkDataMetaDataDescriptor

Upstream requirements: RS_Diag_04266

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_bulk_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdBulkData`
Symbol:	BulkDataMetaDataDescriptor
Syntax:	`struct BulkDataMetaDataDescriptor {...};`
Description:	Definition of bulk data Meta Data Descriptor.

⌋

8.51.2.1 Public Member Variables

8.51.2.1.1 creation_date

[SWS_DM_01837] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::creation_date

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	creation_date
Type:	`ara::core::String`
Syntax:	`ara::core::String creation_date;`
Description:	Specifies the bulk data creation_date.

⌋

8.51.2.1.2 file_name

[SWS_DM_01836] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::file_name

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	file_name
Type:	`ara::core::String`
Syntax:	`ara::core::String file_name;`
Description:	Specifies the filename.

⌋

8.51.2.1.3 hash

[SWS_DM_01839] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::hash

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	hash
Type:	`ara::core::String`
Syntax:	`ara::core::String hash;`
Description:	Specifies the bulk data hash.

⌋

8.51.2.1.4 hash_algorithm

[SWS_DM_01840] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::hash_algorithm

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	hash_algorithm
Type:	`ara::core::String`
Syntax:	`ara::core::String hash_algorithm;`
Description:	Specifies the bulk data hash_algorithm.

⌋

8.51.2.1.5 id

[SWS_DM_01831] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::id

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	id
Type:	`ara::core::String`
Syntax:	`ara::core::String id;`
Description:	Specifies the bulk data id.

⌋

8.51.2.1.6 last_modified

[SWS_DM_01838] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::last_modified

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	last_modified
Type:	`ara::core::String`
Syntax:	`ara::core::String last_modified;`
Description:	Specifies the bulk data last_modified.

⌋

8.51.2.1.7 mimetype

[SWS_DM_01832] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::mimetype

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	mimetype
Type:	`ara::core::String`
Syntax:	`ara::core::String mimetype;`
Description:	Specifies the bulk data mimetype.

⌋

8.51.2.1.8 name

[SWS_DM_01833] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::name

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	name
Type:	`ara::core::String`
Syntax:	`ara::core::String name;`
Description:	Specifies the bulk data name.

⌋

8.51.2.1.9 size

[SWS_DM_01835] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::size

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	size
Type:	`ara::core::Optional< std::uint64_t >`
Syntax:	`ara::core::Optional<std::uint64_t> size;`
Description:	Specifies the bulk data size in Byte.

⌋

8.51.2.1.10 translation_id

[SWS_DM_01834] Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::translation_id

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor`
Symbol:	translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String translation_id;`
Description:	Specifies the bulk data translation_id.

⌋

8.51.3 Struct: DeleteByCategoryResult

[SWS_DM_01926] Definition of API class ara::diag::SovdBulkData::DeleteByCategoryResult

Upstream requirements: RS_Diag_04266

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_bulk_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdBulkData`
Symbol:	DeleteByCategoryResult
Syntax:	`struct DeleteByCategoryResult {...};`
Description:	Definition of the result type for the "Delete All Bulk Data Defined by Category" method .

⌋

8.51.3.1 Public Member Variables

8.51.3.1.1 deleted_items

[SWS_DM_01842] Definition of API variable ara::diag::SovdBulkData::DeleteBy CategoryResult::deleted_items

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::DeleteByCategoryResult`
Symbol:	deleted_items
Type:	`ara::core::Vector< BulkDataMetaDataDescriptor >`
Syntax:	`ara::core::Vector<BulkDataMetaDataDescriptor> deleted_items;`
Description:	List of bulk data ids that were successfully deleted .

⌋

8.51.3.1.2 failed_items

[SWS_DM_01843] Definition of API variable ara::diag::SovdBulkData::DeleteBy CategoryResult::failed_items

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::DeleteByCategoryResult`
Symbol:	failed_items
Type:	`ara::core::Vector< DeletionError >`
Syntax:	`ara::core::Vector<DeletionError> failed_items;`
Description:	A list of DeletionErrors that contains the bulk data instances with corresponding deletion errors .

⌋

8.51.4 Struct: DeletionError

[SWS_DM_01844] Definition of API class ara::diag::SovdBulkData::DeletionError

Upstream requirements: RS_Diag_04266

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_bulk_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdBulkData`
Symbol:	DeletionError
Syntax:	`struct DeletionError {...};`
Description:	Definition of DeletionError for a single bulk-data instance .

⌋

8.51.4.1 Public Member Variables

8.51.4.1.1 error

[SWS_DM_01846] Definition of API variable ara::diag::SovdBulkData::DeletionError::error

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::DeletionError`
Symbol:	error
Type:	`ara::core::ErrorCode`
Syntax:	`ara::core::ErrorCode error;`
Description:	Corresponding error, that specifies why the bulk data could not be deleted .

⌋

8.51.4.1.2 item

[SWS_DM_01845] Definition of API variable ara::diag::SovdBulkData::DeletionError::item

Upstream requirements: RS_Diag_04266

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	`struct ara::diag::SovdBulkData::DeletionError`
Symbol:	item
Type:	`BulkDataMetaDataDescriptor`
Syntax:	`BulkDataMetaDataDescriptor item;`
Description:	bulk data that could not be deleted

⌋

8.52 Header: ara/diag/sovd_configuration.h

8.52.1 Class: SovdConfiguration

[SWS_DM_01860] Definition of API class ara::diag::SovdConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_configuration.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdConfiguration
Syntax:	`class SovdConfiguration {...};`
Description:	Generic SOVD Configuration interface.

⌋

8.52.1.1 Public Member Functions

8.52.1.1.1 Special Member Functions

8.52.1.1.1.1 Destructor

[SWS_DM_01865] Definition of API function ara::diag::SovdConfiguration::~SovdConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`class ara::diag::SovdConfiguration`
Syntax:	`virtual ~SovdConfiguration () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdConfiguration .

⌋

8.52.1.1.2 Constructors

8.52.1.1.2.1 SovdConfiguration

[SWS_DM_01864] Definition of API function ara::diag::SovdConfiguration::SovdConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	`class ara::diag::SovdConfiguration`	
Syntax:	`SovdConfiguration (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a `DiagnosticSovdConfigurationInterface`
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	InstanceSpecifierMappingIntegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMappingViolation	A PortPrototype that is referenced by a DiagnosticSovdConfigurationPortMapping needs to be typed by a DiagnosticSovdConfigurationInterface.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for the SovdConfiguration Interface.	

⌋

8.52.1.1.3 Member Functions

8.52.1.1.3.1 Offer

[SWS_DM_01872] Definition of API function ara::diag::SovdConfiguration::Offer

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	class ara::diag::SovdConfiguration	
Syntax:	ara::core::Result< void > Offer () noexcept;	
Return value:	ara::core::Result< void >	Positive result if service was offered, error if offer failed
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		if handle is an invalid handle
	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		if service has been offered already
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.52.1.1.3.2 RequestGetConfiguration

[SWS_DM_01870] Definition of API function ara::diag::SovdConfiguration::RequestGetConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	`class ara::diag::SovdConfiguration`	
Syntax:	`virtual ara::core::Future< std::tuple< DataTransferReadSession, SovdConfigurationMetaInfo > > RequestGetConfiguration (ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	configurationId	configuration-id of the configuration instance to be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< std::tuple< DataTransferReadSession, SovdConfigurationMetaInfo > >	A future with either an instance of a "SOVD file reading session" with corresponding SovdConfigurationMetaInfo or a SOVD Error -
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Read Configuration".	

⌋

8.52.1.1.3.3 RequestPutConfiguration

[SWS_DM_01871] Definition of API function ara::diag::SovdConfiguration::RequestPutConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	`class ara::diag::SovdConfiguration`	
Syntax:	`virtual ara::core::Future< std::tuple< DataTransferWriteSession, SovdConfigurationMetaInfo > > RequestPutConfiguration (ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	configurationId	configuration-id of the configuration instance to be written
	metaInfo	Contains additional meta information

▽

△

	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< std::tuple< DataTransfer WriteSession, Sovd ConfigurationMetaInfo > >	A future with either an instance of a "SOVD file writing session" with corresponding SovdConfigurationMetaInfo or an SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Write Configuration".	

⌋

8.52.1.1.3.4 StopOffer

[SWS_DM_01930] Definition of API function ara::diag::SovdConfiguration::StopOffer

Upstream requirements: RS_Diag_04261

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`class ara::diag::SovdConfiguration`
Syntax:	`void StopOffer () const noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

⌋

8.52.1.2 Protected Member Functions

8.52.1.2.1 Special Member Functions

8.52.1.2.1.1 Copy Constructor

[SWS_DM_01866] Definition of API function ara::diag::SovdConfiguration::SovdConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`class ara::diag::SovdConfiguration`
Syntax:	`SovdConfiguration (SovdConfiguration const &)=delete;`
Description:	SovdConfiguration shall be a single not copy-able instance.
Visibility:	protected

⌋

8.52.1.2.1.2 Move Constructor

[SWS_DM_01867] Definition of API function ara::diag::SovdConfiguration::SovdConfiguration

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	`class ara::diag::SovdConfiguration`	
Syntax:	`SovdConfiguration (SovdConfiguration &&other) noexcept;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move constructs an instance of SovdConfiguration.	
Visibility:	protected	

⌋

8.52.1.2.1.3 Copy Assignment Operator

[SWS_DM_01868] Definition of API function ara::diag::SovdConfiguration::operator=

Upstream requirements: RS_Diag_04261

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`class ara::diag::SovdConfiguration`
Syntax:	`SovdConfiguration & operator= (SovdConfiguration const &)=delete;`
Description:	SovdConfiguration shall be a single not assignable instance.
Visibility:	protected

⌋

8.52.1.2.1.4 Move Assignment Operator

[SWS_DM_01869] Definition of API function ara::diag::SovdConfiguration::operator=

Upstream requirements: RS_Diag_04261

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	`class ara::diag::SovdConfiguration`	
Syntax:	`SovdConfiguration & operator= (SovdConfiguration &&other) & noexcept;`	
Parameters (out):	other	The other object
Return value:	SovdConfiguration &	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of SovdConfiguration.	
Visibility:	protected	

⌋

8.52.2 Struct: SovdConfigurationMetaInfo

[SWS_DM_01861] Definition of API class ara::diag::SovdConfiguration::SovdConfigurationMetaInfo

Upstream requirements: RS_Diag_04261

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_configuration.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdConfiguration`
Symbol:	SovdConfigurationMetaInfo
Syntax:	`struct SovdConfigurationMetaInfo {...};`
Description:	MetaInfo for Transfer of SOVD Configurations.

⌋

8.52.2.1 Public Member Variables

8.52.2.1.1 mimetype

[SWS_DM_01863] Definition of API variable ara::diag::SovdConfiguration::SovdConfigurationMetaInfo::mimetype

Upstream requirements: RS_Diag_04261

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`struct ara::diag::SovdConfiguration::SovdConfigurationMetaInfo`
Symbol:	mimetype
Type:	`ara::core::String`
Syntax:	`ara::core::String mimetype;`
Description:	Specifies the configuration mimetype .

⌋

8.52.2.1.2 size

[SWS_DM_01862] Definition of API variable ara::diag::SovdConfiguration::SovdConfigurationMetaInfo::size

Upstream requirements: RS_Diag_04261

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_configuration.h"
Scope:	`struct ara::diag::SovdConfiguration::SovdConfigurationMetaInfo`
Symbol:	size
Type:	`ara::core::Optional< std::uint64_t >`
Syntax:	`ara::core::Optional<std::uint64_t> size;`
Description:	Specifies the size of the configuration data .

⌋

8.53 Header: ara/diag/sovd_http_redirect.h

8.53.1 Struct: HttpRedirect

[SWS_DM_01828] Definition of API class ara::diag::HttpRedirect

Upstream requirements: RS_Diag_04266, RS_Diag_04256

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_http_redirect.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	HttpRedirect
Syntax:	`struct HttpRedirect {...};`
Description:	Definition of Http Redirect. In case a SOVD resource has been moved and/or is aviable under another URI, the SOVD server shall use this type to indicate the new location of the redirected SOVD resource to the SOVD client.

⌋

8.53.1.1 Public Member Variables

8.53.1.1.1 url

[SWS_DM_01829] Definition of API variable ara::diag::HttpRedirect::url

Upstream requirements: RS_Diag_04266, RS_Diag_04256

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_http_redirect.h"
Scope:	`struct ara::diag::HttpRedirect`
Symbol:	url
Type:	`ara::core::String`
Syntax:	`ara::core::String url;`
Description:	Specifies the url where the request shall be redirected.

⌋

8.54 Header: ara/diag/sovd_proximity_challenge.h

8.54.1 Class: SovdProximityChallenge

[SWS_DM_01481] Definition of API class ara::diag::SovdProximityChallenge

Upstream requirements: RS_Diag_04262

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdProximityChallenge
Syntax:	`class SovdProximityChallenge {...};`
Description:	Class to implement the SOVD Proximity Challenge interfaces in the application.

⌋

8.54.1.1 Public Member Functions

8.54.1.1.1 Special Member Functions

8.54.1.1.1.1 Destructor

[SWS_DM_01479] Definition of API function ara::diag::SovdProximityChallenge::~SovdProximityChallenge

Upstream requirements: RS_Diag_04262

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	`class ara::diag::SovdProximityChallenge`
Syntax:	`virtual ~SovdProximityChallenge () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdProximityChallenge .

⌋

8.54.1.1.2 Constructors

8.54.1.1.2.1 SovdProximityChallenge

[SWS_DM_01480] Definition of API function ara::diag::SovdProximityChallenge::SovdProximityChallenge

Upstream requirements: RS_Diag_04262

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Scope:	`class ara::diag::SovdProximityChallenge`	
Syntax:	`explicit SovdProximityChallenge (ara::core::InstanceSpecifier instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdProximityChallenge service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	InstanceSpecifierMappingIntegrityViolation	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	PortInterfaceMappingViolation	A PortPrototype that is referenced by a DiagnosticSovdProximityChallengePortMapping needs to be typed by a DiagnosticSovdProximityChallengeInterface.
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	InstanceSpecifierAlreadyInUseViolation	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructs the port for SOVD Proximity Challenge interface.	

⌋

8.54.1.1.3 Member Functions

8.54.1.1.3.1 GetChallenge

[SWS_DM_01494] Definition of API function ara::diag::SovdProximityChallenge::GetChallenge

Upstream requirements: RS_Diag_04262

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Scope:	class ara::diag::SovdProximityChallenge	
Syntax:	virtual ara::core::Future< SovdProximityChallengeType > GetChallenge (const MetaInfo &metaInfo) noexcept=0;	
Parameters (in):	metaInfo	The meta information of a diagnostic service port
Return value:	ara::core::Future< SovdProximityChallengeType >	Challenge created by the application and point in time until when the challenge is valid (valid_until)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the ara::diag::ConcurrencyType given in the constructor
Description:	This function returns a proximity challenge for the client.	

⌋

8.54.1.1.3.2 Offer

[SWS_DM_01478] Definition of API function ara::diag::SovdProximityChallenge::Offer

Upstream requirements: RS_Diag_04262

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Scope:	`class ara::diag::SovdProximityChallenge`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics
		Implementation does not fit to the configuration.
Description:	This Offer will enable the DM to forward request messages to this handler.	

⌋

8.54.1.1.3.3 StopOffer

[SWS_DM_01495] Definition of API function ara::diag::SovdProximityChallenge::StopOffer

Upstream requirements: RS_Diag_04262

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	`class ara::diag::SovdProximityChallenge`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM .

⌋

8.54.1.1.3.4 ValidateResponse

[SWS_DM_01493] Definition of API function ara::diag::SovdProximityChallenge::ValidateResponse

Upstream requirements: RS_Diag_04262

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Scope:	`class ara::diag::SovdProximityChallenge`	
Syntax:	`virtual ara::core::Future< void > ValidateResponse (ara::core::Span< const ara::core::Byte > proximityResponse, const MetaInfo &metaInfo) noexcept=0;`	
Parameters (in):	proximityResponse	The proximity_response sent by the client, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	metaInfo	The meta information of a diagnostic service port
Return value:	ara::core::Future< void >	Value if the response was valid
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	This function validates the proximity response sent by the client.	

⌋

8.54.2 Struct: SovdProximityChallengeType

[SWS_DM_01482] Definition of API class ara::diag::SovdProximityChallenge Type

Upstream requirements: RS_Diag_04262

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdProximityChallengeType
Syntax:	`struct SovdProximityChallengeType {...};`
Description:	Holds the data returned by the SovdProximityChallenge::GetChallenge()

⌋

8.54.2.1 Public Member Variables

8.54.2.1.1 challenge

[SWS_DM_01492] Definition of API variable ara::diag::SovdProximityChallenge Type::challenge

Upstream requirements: RS_Diag_04262

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	`struct ara::diag::SovdProximityChallengeType`
Symbol:	challenge
Type:	`ara::core::Span< ara::core::Byte >`
Syntax:	`ara::core::Span<ara::core::Byte> challenge;`
Description:	Challenge which has to be solved by the SOVD client to prove proximity.

⌋

8.54.2.1.2 valid_until

[SWS_DM_01491] Definition of API variable ara::diag::SovdProximityChallenge Type::valid_until

Upstream requirements: RS_Diag_04262

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	`struct ara::diag::SovdProximityChallengeType`
Symbol:	valid_until
Type:	`std::chrono::system_clock::time_point`
Syntax:	`std::chrono::system_clock::time_point valid_until;`
Description:	The time until the challenge is valid.

⌋

8.55 Header: ara/diag/sovd_service_validation.h

8.55.1 Non-Member Types

8.55.1.1 Enumeration: SovdRequestMethod

[SWS_DM_01821] Definition of API enum ara::diag::SovdRequestMethod

Upstream requirements: RS_Diag_04256, RS_Diag_04273

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/sovd_service_validation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	SovdRequestMethod	
Underlying type:	--	
Syntax:	`enum class SovdRequestMethod {...};`	
Values:	kGet	--
	kPut	< GET.PUT.
	kPost	POST.
	kDelete	DELETE.
Description:	HTTP method used in the SOVD request.	

⌋

8.55.2 Class: SovdServiceValidation

[SWS_DM_01822] Definition of API class ara::diag::SovdServiceValidation

Upstream requirements: RS_Diag_04256

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_service_validation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdServiceValidation
Syntax:	`class SovdServiceValidation {...};`
Description:	Class to implement manufacturer-/supplier-specific checks for SOVD in the application.

⌋

8.55.2.1 Public Member Functions

8.55.2.1.1 Special Member Functions

8.55.2.1.1.1 Destructor

[SWS_DM_01824] Definition of API function ara::diag::SovdServiceValidation::~SovdServiceValidation

Upstream requirements: RS_Diag_04256

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_service_validation.h"
Scope:	`class ara::diag::SovdServiceValidation`
Syntax:	`virtual ~SovdServiceValidation () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdServiceValidation .

⌋

8.55.2.1.2 Constructors

8.55.2.1.2.1 SovdServiceValidation

[SWS_DM_01823] Definition of API function ara::diag::SovdServiceValidation::SovdServiceValidation

Upstream requirements: RS_Diag_04256

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_service_validation.h"	
Scope:	`class ara::diag::SovdServiceValidation`	
Syntax:	`explicit SovdServiceValidation (ara::core::InstanceSpecifier instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a `DiagnosticSovdServiceValidationInterface` service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticSovdServiceValidationPortMapping` needs to be typed by a `DiagnosticSovdServiceValidationInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructs the port for SovdServiceValidation interface.	

⌋

8.55.2.1.3 Member Functions

8.55.2.1.3.1 Offer

[SWS_DM_01825] Definition of API function ara::diag::SovdServiceValidation::Offer

Upstream requirements: RS_Diag_04256

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_service_validation.h"	
Scope:	`class ara::diag::SovdServiceValidation`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	--
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		This service was already offered.
	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics
		Implementation does not fit to the configuration.
Description:	Enable the DM to forward request messages to this handler.	

⌋

8.55.2.1.3.2 StopOffer

[SWS_DM_01826] Definition of API function ara::diag::SovdServiceValidation::StopOffer

Upstream requirements: RS_Diag_04256

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_service_validation.h"
Scope:	`class ara::diag::SovdServiceValidation`
Syntax:	`void StopOffer () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Disable the forwarding of request messages from DM .

⌋

8.55.2.1.3.3 Validate

[SWS_DM_01827] Definition of API function ara::diag::SovdServiceValidation::Validate

Upstream requirements: RS_Diag_04256, RS_Diag_04273

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_service_validation.h"	
Scope:	`class ara::diag::SovdServiceValidation`	
Syntax:	`virtual ara::core::Future< void > Validate (SovdRequestMethod requestMethod, const MetaInfo &metaInfo) noexcept;`	
Parameters (in):	requestMethod	The HTTP method used in the SOVD request message.
	metaInfo	MetaInfo of the request (includes the path of the requested resource or resource collection).
Return value:	ara::core::Future< void >	Value if the request is allowed, or an error.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Errors:	DiagSovdErrc::kSovdServerMisconfigured	rollback_semantics
		If there is some fatal error in the setup of the application.
	DiagSovdErrc::kPreconditionNotFulfilled	rollback_semantics
		If the precondition for the accessing the resource is not fulfilled.

▽

△

	DiagSovdErrc::kTemporarilyNotAvailable	rollback_semantics
		If the request currently can't be served.
	DiagSovdErrc::kInsufficientAccessRights	rollback_semantics
		Or any other DiagSovdErrc for an SOVD-specific error message.
	DiagUdsNrcErrc::kRpmTooHigh	rollback_semantics
		Or any other DiagUdsNrcErrc in case this is an error standardized in UDS.
Description:	Called for every SOVD request message. Will be called multiple times if the request is for a resource collection: • once for the resource collection, • once for each resource in the resource collection.	

⌋

8.56 Header: ara/diag/sovd_sw_update.h

8.56.1 Non-Member Types

8.56.1.1 Enumeration: SovdUpdatePhase

[SWS_DM_01875] Definition of API enum ara::diag::SovdUpdatePhase

Upstream requirements: RS_Diag_04265

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	SovdUpdatePhase	
Underlying type:	std::uint8_t	
Syntax:	`enum class SovdUpdatePhase : std::uint8_t {...};`	
Values:	kPrepare	--
	kExecute	< The Update is in preparation.< The Update is in execution
Description:	Phase type for a SOVD update.	

⌋

8.56.1.2 Enumeration: SovdUpdateStatus

[SWS_DM_01876] Definition of API enum ara::diag::SovdUpdateStatus

Upstream requirements: RS_Diag_04265

⌈

Kind:	enumeration	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	`namespace ara::diag`	
Symbol:	SovdUpdateStatus	
Underlying type:	std::uint8_t	
Syntax:	`enum class SovdUpdateStatus : std::uint8_t {...};`	
Values:	kPending	The current Update phase is pending.
	kInProgress	The current Update phase is inProgress.
	kFailed	The current Update phase has failed.
	kCompleted	The current Update phase has been completed.
Description:	Status type for a SOVD update .	

⌋

8.56.2 Class: SovdSwUpdate

[SWS_DM_01874] Definition of API class ara::diag::SovdSwUpdate

Upstream requirements: RS_Diag_04265

⌈

Kind:	class
Header file:	#include "ara/diag/sovd_sw_update.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`namespace ara::diag`
Symbol:	SovdSwUpdate
Syntax:	`class SovdSwUpdate {...};`
Description:	Class to implement the SovdSwUpdate interface.

⌋

8.56.2.1 Public Member Functions

8.56.2.1.1 Special Member Functions

8.56.2.1.1.1 Destructor

[SWS_DM_01911] Definition of API function ara::diag::SovdSwUpdate::~SovdSwUpdate

Upstream requirements: RS_Diag_04265

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`class ara::diag::SovdSwUpdate`
Syntax:	`virtual ~SovdSwUpdate () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class SovdSwUpdate .

⌋

8.56.2.1.2 Constructors

8.56.2.1.2.1 SovdSwUpdate

[SWS_DM_01910] Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`SovdSwUpdate (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;`	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a `DiagnosticSovdUpdateInterface` service instance in the manifest
	concurrencyType	Specifies if the interface is implemented as thread-safe(kConcurrent) or non-thread safe (kNonConcurrent)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype. String format: "Violation detected in {process Identifier} at {location}: Invalid InstanceSpecifer {instanceSpecifier} in a constructor of class: {className}"
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `DiagnosticSovdUpdatePortMapping` needs to be typed by a `DiagnosticSovdUpdateInterface`.
	`ProcessMappingViolation`	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process. String format: "Violation detected in {processIdentifier} at {location}: Invalid Instance Specifer {instanceSpecifier} in a constructor of class: {className}"
	`InstanceSpecifierAlreadyInUseViolation`	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process. String format: "Violation detected in {process Identifier} at {location}: InstanceSpecifer {instanceSpecifier} in constructor of class {className} already in use in this process"
Description:	Constructor for SovdSwUpdate.	

⌋

8.56.2.1.3 Member Functions

8.56.2.1.3.1 DeleteUpdatePackage

[SWS_DM_01922] Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< void > DeleteUpdatePackage (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id to be deleted
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< void >	A future with either a positive results or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Delete Update Package from an SOVD Server".	

⌋

8.56.2.1.3.2 ExecuteUpdatePackage

[SWS_DM_01920] Definition of API function ara::diag::SovdSwUpdate::ExecuteUpdatePackage

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< void > ExecuteUpdatePackage (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id of the update to be executed
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< void >	A future with either a positive results or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Execute Installation of an Update".	

⌋

8.56.2.1.3.3 GetAllUpdates

[SWS_DM_01916] Definition of API function ara::diag::SovdSwUpdate::GetAllUpdates

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< ara::core::Vector< ara::core::String > > GetAllUpdates (ara::core::String targetVersion, ara::core::String origin, MetaInfo const &metaInfo, CancellationHandler cancellation Handler) noexcept=0;`	
Parameters (in):	targetVersion	Optional parameter to be used if the target-version query parameter is used in the request. If the parameter is not set the String shall be left empty. set in the request the String shall be left empty
	origin	Optional parameter to be used if the origin query parameter is used in the request. If the parameter is not set the String shall be left empty

▽

△

	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< ara::core::Vector< ara::core::String > >	A future with a span of all update package ids. The span shall not be modified until the releaseHandler is called.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Retrieve List of All Updates".	

⌋

8.56.2.1.3.4 GetUpdatePackageDetails

[SWS_DM_01917] Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackageDetails

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< UpdatePackageDetails > GetUpdatePackageDetails (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id of the update to be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< UpdatePackageDetails >	A future with the details of the requested update package or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Get Details of Update".	

⌋

8.56.2.1.3.5 GetUpdatePackageStatus

[SWS_DM_01921] Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackageStatus

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< UpdatePackageStatus > GetUpdatePackageStatus (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id of which the status shall be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< UpdatePackageStatus >	A future with either the current UpdatePackageStatus or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Get Status of an Update".	

⌋

8.56.2.1.3.6 Offer

[SWS_DM_01924] Definition of API function ara::diag::SovdSwUpdate::Offer

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`ara::core::Result< void > Offer () noexcept;`	
Return value:	ara::core::Result< void >	Positive result if service was offered, error if offer failed
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		if handle is an invalid handle
	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		if service has been offered already

▽

△

Description:	This Offer will enable the DM to forward request messages to this handler.

⌋

8.56.2.1.3.7 PrepareUpdatePackage

[SWS_DM_01919] Definition of API function ara::diag::SovdSwUpdate::PrepareUpdatePackage

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< void > PrepareUpdatePackage (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id of the update to be prepared
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< void >	A future with either a positive result or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Prepare Installation of an Update".	

⌋

8.56.2.1.3.8 PutUpdatePackageAutomated

[SWS_DM_01918] Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackageAutomated

Upstream requirements: RS_Diag_04265

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`class ara::diag::SovdSwUpdate`

▽

△

Syntax:	`virtual ara::core::Future< void > PutUpdatePackageAutomated (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	updatePackageId	update-package-id of the update to be put to automated
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< void >	A future with either a positive result or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Automated Installation of an Update".	

⌋

8.56.2.1.3.9 RequestUpdatePackageRegistration

[SWS_DM_01923] Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`virtual ara::core::Future< std::tuple< DataTransferWriteSession, ara::core::String > > RequestUpdatePackageRegistration (MetaInfo const &metaInfo, ara::core::Optional< uint64_t > expectedSize, CancellationHandler cancellationHandler) noexcept=0;`	
Parameters (in):	metaInfo	Contains additional meta information
	expectedSize	Specifies the expected size if available in the request
	cancellationHandler	Informs if the current conversation is canceled
Return value:	ara::core::Future< std::tuple< DataTransferWriteSession, ara::core::String > >	A future with either an instance of a "SOVD file writing session" with the corresponding update package id or a SOVD Error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread-safe or not is determined by the `ara::diag::ConcurrencyType` given in the constructor
Description:	Called for SOVD method "Register an Update at the SOVD Server".	

⌋

8.56.2.1.3.10 StopOffer

[SWS_DM_01925] Definition of API function ara::diag::SovdSwUpdate::StopOffer

Upstream requirements: RS_Diag_04265

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`class ara::diag::SovdSwUpdate`
Syntax:	`void StopOffer () const noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

⌋

8.56.2.2 Protected Member Functions

8.56.2.2.1 Special Member Functions

8.56.2.2.1.1 Copy Constructor

[SWS_DM_01912] Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate

Upstream requirements: RS_Diag_04265

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`class ara::diag::SovdSwUpdate`
Syntax:	`SovdSwUpdate (SovdSwUpdate const &)=delete;`
Description:	SovdSwUpdate shall be a single not copy-able instance.
Visibility:	protected

⌋

8.56.2.2.1.2 Move Constructor

[SWS_DM_01913] Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`SovdSwUpdate (SovdSwUpdate &&other) noexcept;`	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	unsafe	
Description:	Move constructs an instance of SovdSwUpdate.	
Visibility:	protected	

⌋

8.56.2.2.1.3 Copy Assignment Operator

[SWS_DM_01914] Definition of API function ara::diag::SovdSwUpdate::operator=

Upstream requirements: RS_Diag_04265

⌈

Kind:	function
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`class ara::diag::SovdSwUpdate`
Syntax:	`SovdSwUpdate & operator= (SovdSwUpdate const &)=delete;`
Description:	SovdSwUpdate shall be a single not assignable instance.
Visibility:	protected

⌋

8.56.2.2.1.4 Move Assignment Operator

[SWS_DM_01915] Definition of API function ara::diag::SovdSwUpdate::operator=

Upstream requirements: RS_Diag_04265

⌈

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	`class ara::diag::SovdSwUpdate`	
Syntax:	`SovdSwUpdate & operator= (SovdSwUpdate &&other) & noexcept;`	
Parameters (out):	other	The other object
Return value:	SovdSwUpdate &	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of SovdSwUpdate.	
Visibility:	protected	

⌋

8.56.3 Struct: SubProgress

[SWS_DM_01896] Definition of API class ara::diag::SovdSwUpdate::SubProgress

Upstream requirements: RS_Diag_04265

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_sw_update.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdSwUpdate`
Symbol:	SubProgress
Syntax:	`struct SubProgress {...};`
Description:	Definition of Sovd Progress type.

⌋

8.56.3.1 Public Member Variables

8.56.3.1.1 entity

[SWS_DM_01898] Definition of API variable ara::diag::SovdSwUpdate::SubProgress::entity

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::SubProgress`
Symbol:	entity
Type:	`ara::core::String`
Syntax:	`ara::core::String entity;`
Description:	Specifies the entity that is currently being updated.

⌋

8.56.3.1.2 error

[SWS_DM_01901] Definition of API variable ara::diag::SovdSwUpdate::SubProgress::error

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::SubProgress`
Symbol:	error
Type:	`ara::core::ErrorCode`
Syntax:	`ara::core::ErrorCode error;`
Description:	Specifies the error if status is failed.

⌋

8.56.3.1.3 progress

[SWS_DM_01900] Definition of API variable ara::diag::SovdSwUpdate::SubProgress::progress

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::SubProgress`
Symbol:	progress
Type:	`ara::core::Optional< std::uint8_t >`
Syntax:	`ara::core::Optional<std::uint8_t> progress;`
Description:	Specifies the Update progress in percent.

⌋

8.56.3.1.4 status

[SWS_DM_01899] Definition of API variable ara::diag::SovdSwUpdate::SubProgress::status

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::SubProgress`
Symbol:	status
Type:	`SovdUpdateStatus`
Syntax:	`SovdUpdateStatus status;`
Description:	Specifies the Update Status.

⌋

8.56.4 Struct: UpdatePackageDetails

[SWS_DM_01877] Definition of API class ara::diag::SovdSwUpdate::UpdatePackageDetails

Upstream requirements: RS_Diag_04265

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_sw_update.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdSwUpdate`
Symbol:	UpdatePackageDetails
Syntax:	`struct UpdatePackageDetails {...};`
Description:	Definition of Update Package Details .

⌋

8.56.4.1 Public Member Variables

8.56.4.1.1 affected_components

[SWS_DM_01895] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::affected_components

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	affected_components
Type:	`ara::core::Vector< ara::core::String >`
Syntax:	`ara::core::Vector<ara::core::String> affected_components;`
Description:	Specifies the update package affected_components.

⌋

8.56.4.1.2 automated

[SWS_DM_01882] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::automated

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	automated
Type:	`ara::core::Optional< bool >`
Syntax:	`ara::core::Optional<bool> automated;`
Description:	Specifies the update package attribute "automated".

⌋

8.56.4.1.3 duration

[SWS_DM_01891] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::duration

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	duration
Type:	`ara::core::Optional< std::uint64_t >`
Syntax:	`ara::core::Optional<std::uint64_t> duration;`
Description:	Specifies the update package duration in seconds.

⌋

8.56.4.1.4 execution_conditions

[SWS_DM_01890] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::execution_conditions

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	execution_conditions
Type:	`ara::core::String`
Syntax:	`ara::core::String execution_conditions;`
Description:	Specifies the update package execution_conditions.

⌋

8.56.4.1.5 id

[SWS_DM_01878] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	id
Type:	`ara::core::String`
Syntax:	`ara::core::String id;`
Description:	Specifies the update package id.

⌋

8.56.4.1.6 name

[SWS_DM_01879] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::name

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	name
Type:	`ara::core::String`
Syntax:	`ara::core::String name;`
Description:	Specifies the update package update_name.

⌋

8.56.4.1.7 notes

[SWS_DM_01884] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::notes

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	notes
Type:	`ara::core::String`
Syntax:	`ara::core::String notes;`
Description:	Specifies the update package notes.

⌋

8.56.4.1.8 notes_translation_id

[SWS_DM_01885] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::notes_translation_id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	notes_translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String notes_translation_id;`
Description:	Specifies the update package notes_translation_id.

⌋

8.56.4.1.9 origin

[SWS_DM_01883] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::origin

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	origin
Type:	`ara::core::Vector< ara::core::String >`
Syntax:	`ara::core::Vector<ara::core::String> origin;`
Description:	Specifies the update package origin.

⌋

8.56.4.1.10 preconditions

[SWS_DM_01888] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::preconditions

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	preconditions
Type:	`ara::core::String`
Syntax:	`ara::core::String preconditions;`
Description:	Specifies the update package preconditions.

⌋

8.56.4.1.11 preconditions_translation_id

[SWS_DM_01889] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::preconditions_translation_id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	preconditions_translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String preconditions_translation_id;`
Description:	Specifies the update package preconditions_translation_id.

⌋

8.56.4.1.12 size

[SWS_DM_01893] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::size

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	size
Type:	`std::uint64_t`
Syntax:	`std::uint64_t size;`
Description:	Specifies the update package size in Bytes .

⌋

8.56.4.1.13 translation_id

[SWS_DM_01881] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::translation_id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String translation_id;`
Description:	Specifies the update package update_translation_id.

⌋

8.56.4.1.14 updated_components

[SWS_DM_01894] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::updated_components

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	updated_components
Type:	`ara::core::Vector< ara::core::String >`
Syntax:	`ara::core::Vector<ara::core::String> updated_components;`
Description:	Specifies the update package updated_components.

⌋

8.56.4.1.15 user_activity

[SWS_DM_01886] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::user_activity

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	user_activity
Type:	`ara::core::String`
Syntax:	`ara::core::String user_activity;`
Description:	Specifies the update package user_activity.

⌋

8.56.4.1.16 user_activity_translation_id

[SWS_DM_01887] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::user_activity_translation_id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageDetails`
Symbol:	user_activity_translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String user_activity_translation_id;`
Description:	Specifies the update package user_activity_translation_id.

⌋

8.56.5 Struct: UpdatePackageStatus

[SWS_DM_01902] Definition of API class ara::diag::SovdSwUpdate::UpdatePackageStatus

Upstream requirements: RS_Diag_04265

⌈

Kind:	struct
Header file:	#include "ara/diag/sovd_sw_update.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	`class ara::diag::SovdSwUpdate`
Symbol:	UpdatePackageStatus
Syntax:	`struct UpdatePackageStatus {...};`
Description:	Definition of Update Status.

⌋

8.56.5.1 Public Member Variables

8.56.5.1.1 error

[SWS_DM_01909] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::error

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	error
Type:	`ara::core::Optional< ara::diag::DiagSovdErrc >`
Syntax:	`ara::core::Optional<ara::diag::DiagSovdErrc> error;`
Description:	Specifies the error if status is failed.

⌋

8.56.5.1.2 phase

[SWS_DM_01903] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::phase

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	phase
Type:	`SovdUpdatePhase`
Syntax:	`SovdUpdatePhase phase;`
Description:	Specifies the Update Status phase.

⌋

8.56.5.1.3 progress

[SWS_DM_01905] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::progress

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	progress
Type:	`ara::core::Optional< std::uint8_t >`
Syntax:	`ara::core::Optional<std::uint8_t> progress;`
Description:	Specifies the progress of the Update phase in percentage.

⌋

8.56.5.1.4 status

[SWS_DM_01904] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::status

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	status
Type:	`SovdUpdateStatus`
Syntax:	`SovdUpdateStatus status;`
Description:	Specifies the status of the Update phase.

⌋

8.56.5.1.5 step

[SWS_DM_01907] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	step
Type:	`ara::core::String`
Syntax:	`ara::core::String step;`
Description:	Specifies the current step of the Update.

⌋

8.56.5.1.6 step_translation_id

[SWS_DM_01908] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step_translation_id

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	step_translation_id
Type:	`ara::core::String`
Syntax:	`ara::core::String step_translation_id;`
Description:	Specifies the step_translation_id.

⌋

8.56.5.1.7 subprogress

[SWS_DM_01906] Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::subprogress

Upstream requirements: RS_Diag_04265

⌈

Kind:	variable
Header file:	#include "ara/diag/sovd_sw_update.h"
Scope:	`struct ara::diag::SovdSwUpdate::UpdatePackageStatus`
Symbol:	subprogress
Type:	`ara::core::Vector< SubProgress >`
Syntax:	`ara::core::Vector<SubProgress> subprogress;`
Description:	Specifies the progress of the Update phase in percent.

⌋

9 Service Interfaces

This functional cluster does not define any provided or required service interfaces.

10 Configuration

The configuration model of this functional cluster is defined in [11]. This chapter defines the default values for attributes and semantic constraints for elements specified in [11] that are part of the configuration model of this functional cluster.

The configuration is realized in huge parts using the Autosar Diagnostic Extract Template [3].

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [11].

10.2 Semantic Constraints

This section defines semantic constraints for elements specified in [11] that are part of the configuration model of this functional cluster.

[SWS_DM_CONSTR_00397] Configurable Namespace for Diagnostic Management

Upstream requirements: RS_Diag_04169

⌈`DiagnosticPortInterface.namespace` may only exist for the following `DiagnosticPortInterface`:

- `DiagnosticRoutineInterface`
- `DiagnosticDataIdentifierInterface`
- `DiagnosticDataElementInterface`

⌋

A Mentioned Manifest Elements

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

This chapter is generated.

Class	**ApApplicationError**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform **Tags:** atp.recommendedPackage=ApplicationErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error.
errorDomain	ApApplicationError Domain	0..1	ref	This reference represents the error domain of the Ap ApplicationError.

Table A.1: ApApplicationError

Class	**ArgumentDataPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ClientServerOperation.argument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table A.2: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.

▽

△

Enumeration	**ArgumentDirectionEnum**
Aggregated by	ArgumentDataPrototype.direction, SwServiceArg.direction
Literal	***Description***
in	The argument value is passed to the callee. **Tags:** atp.EnumerationLiteralIndex=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=1
out	The argument value is passed from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=2

Table A.3: ArgumentDirectionEnum

Class	***AutosarDataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, Field, ParameterDataPrototype, PersistencyDataElement, VariableDataPrototype			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table A.4: AutosarDataPrototype

Class	***BaseType*** (abstract)			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependent base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	SwBaseType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table A.5: BaseType

Class	**BaseTypeDirectDefinition**
Package	M2::MSR::AsamHdo::BaseTypes
Note	This BaseType is defined directly (as opposite to a derived BaseType)
Base	*ARObject*, *BaseTypeDefinition*

▽

△

Class	**BaseTypeDirectDefinition**			
Aggregated by	*BaseType*.baseTypeDefinition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:** xml.sequenceOffset=120

Table A.6: BaseTypeDirectDefinition

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, DiagnosticDataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface.write, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, PhmRecoveryActionInterface.recovery, ServiceInterface.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ClientServerOperation**			
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime
fireAndForget	Boolean	0..1	attr	This attribute defines whether this method is a fire&forget method (true) or not (false).
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation.
possibleApError Set	ApApplicationErrorSet	*	ref	This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately.

Table A.7: ClientServerOperation

Class	**CompuConst**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	*ARObject*			
Aggregated by	Compu.compuDefaultValue, CompuScale.compuInverseValue, CompuScaleConstantContents.compu Const			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
compuConst ContentType	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

Table A.8: CompuConst

Class	**CompuConstFormulaContent**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the constant value of the computation method is represented by a variation point. This difference is due to compatibility with ASAM HDO.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**CompuConstFormulaContent**			
vf	Numerical	1	attr	Value calculated via a system constant. This element is included in every case where parameters should be generated from numerical values during compile time (not runtime!). Thus for example, the influence of the cylinder number on conversion formulae can be introduced in a repeatable manner. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

Table A.9: CompuConstFormulaContent

Class	**CompuConstTextContent**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table A.10: CompuConstTextContent

Class	**CompuScale**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Aggregated by	CompuScales.compuScale			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing `CompuMethod` in A2L.
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:** xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** xml.sequenceOffset=30

▽

△

Class	**CompuScale**			
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveUnlimitedInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table A.11: CompuScale

Class	**CompuScaleConstantContents**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that a particular scale of the computation method is constant.			
Base	*ARObject*, *CompuScaleContents*			
Aggregated by	CompuScale.compuScaleContents			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuConst	CompuConst	0..1	aggr	This represents the fact that the scale is a constant. The use case is mainly a non interpolated scale. It is a simplification of the fact that a constant scale can also be expressed as rational function of order 0. **Tags:** xml.sequenceOffset=90

Table A.12: CompuScaleConstantContents

Class	**CompuScales**
Package	M2::MSR::AsamHdo::ComputationMethod
Note	This meta-class represents the ability to stepwise express a computation method.
Base	*ARObject*, *CompuContent*

▽

△

Class	**CompuScales**			
Aggregated by	Compu.compuContent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuScale (ordered)	CompuScale	*	aggr	This represents one scale within the compu method. Note that it contains a Variationpoint in order to support blueprints of enumerations. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=compuScale, compuScale.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

Table A.13: CompuScales

Class	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type.
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType
template Argument (ordered)	CppTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef).

Table A.14: CppImplementationDataType

Class	**DiagEventDebounceCounterBased**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the counter-based debounce algorithm shall be used by the DEM for this diagnostic monitor. This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounce CounterBased.			
Base	*ARObject*, *DiagEventDebounceAlgorithm*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticDebounceAlgorithmProps.debounceAlgorithm, DiagnosticEventNeeds.diagEventDebounce Algorithm			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
counterBased FdcThreshold StorageValue	Integer	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame.
counter DecrementStep Size	Integer	0..1	attr	This value shall be taken to decrement the internal debounce counter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterFailed Threshold	Integer	0..1	attr	This value defines the event-specific limit that indicates the "failed" counter status. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counter IncrementStep Size	Integer	0..1	attr	This value shall be taken to increment the internal debounce counter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterJump Down	Boolean	0..1	attr	This value activates or deactivates the counter jump-down behavior. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterJump DownValue	Integer	0..1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from incrementing to decrementing. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterJumpUp	Boolean	0..1	attr	This value activates or deactivates the counter jump-up behavior. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterJumpUp Value	Integer	0..1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from decrementing to incrementing. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
counterPassed Threshold	Integer	0..1	attr	This value defines the event-specific limit that indicates the "passed" counter status. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.15: DiagEventDebounceCounterBased

Class	**DiagEventDebounceMonitorInternal**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that no Dem pre-debounce algorithm shall be used for this diagnostic monitor. The SWC might implement an internal debouncing algorithm and report qualified (debounced) results to the Dem/DM.			
Base	*ARObject*, *DiagEventDebounceAlgorithm*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticDebounceAlgorithmProps.debounceAlgorithm, DiagnosticEventNeeds.diagEventDebounceAlgorithm			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.16: DiagEventDebounceMonitorInternal

Class	**DiagEventDebounceTimeBased**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the time-based pre-debounce algorithm shall be used by the Dem for this diagnostic monitor. This is related to set the EcuC choice container DemDebounceAlgorithmClass to DemDebounceTimeBase.			
Base	*ARObject*, *DiagEventDebounceAlgorithm*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticDebounceAlgorithmProps.debounceAlgorithm, DiagnosticEventNeeds.diagEventDebounceAlgorithm			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBasedFdcThresholdStorageValue	TimeValue	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
timeFailedThreshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "failed" status. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
timePassedThreshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "passed" status. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.17: DiagEventDebounceTimeBased

Class	***DiagnosticAbstractDataIdentifier*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents an abstract base class for the modeling of a diagnostic data identifier (DID).			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticDataIdentifier, DiagnosticDynamicDataIdentifier			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticAbstractDataIdentifier in the scope of diagnostic workflow **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.18: DiagnosticAbstractDataIdentifier

Class	***DiagnosticAbstractParameter*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents an abstract base class for modeling a diagnostic parameter.			
Base	*ARObject*			
Subclasses	DiagnosticParameter, DiagnosticParameterElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bitOffset	PositiveInteger	0..1	attr	This represents the bitOffset of the DiagnosticParameter. The value of the bitOffset shall always be interpreted as relative to the start of the enclosing DiagnosticData Identifier, DiagnosticParameterIdentifier, or Diagnostic RoutineSubfunction. **Stereotypes:** atpIdentityContributor **Tags:** atp.Status=candidate
dataElement	DiagnosticDataElement	0..1	aggr	This represents the related dataElement of the Diagnostic Parameter **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement.shortName, data Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
parameterSize	PositiveInteger	0..1	attr	This attribute allows for the specification of the parameter size. This information is relevant if there is a gap between one diagnostic parameter and the following diagnostic parameter (or the tail of the telegram). The unit is bit and the values shall be multiples of 8. **Tags:** atp.Status=candidate

Table A.19: DiagnosticAbstractParameter

Class	**DiagnosticAccessPermission**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This represents the specification of whether a given service can be accessed according to the existence of meta-classes referenced by a particular DiagnosticAccessPermission. In other words, this meta-class acts as a mapping element between several (otherwise unrelated) pieces of information that are put into context for the purpose of checking for access rights. **Tags:** atp.recommendedPackage=DiagnosticAccessPermissions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Enabled	DiagnosticAuthRole Proxy	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=authenticationEnabled
diagnostic Session	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticSession
environmental Condition	Diagnostic EnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentalCondition

▽

△

Class	DiagnosticAccessPermission			
securityLevel	DiagnosticSecurityLevel	*	ref	This represents the associated DiagnosticSecurityLevels **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=securityLevel
sovdLock	DiagnosticSovdLock	0..1	ref	This represents the associated SOVD lock. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=sovdLock atp.Status=candidate

Table A.20: DiagnosticAccessPermission

Class	DiagnosticAging			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticAging			
Note	Defines the aging algorithm. **Tags:** atp.recommendedPackage=DiagnosticAgings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
agingCycle	DiagnosticOperation Cycle	0..1	ref	This represents the applicable aging cycle. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=agingCycle.diagnosticOperationCycle, aging Cycle.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
threshold	PositiveInteger	0..1	attr	Number of aging cycles needed to unlearn/delete the event. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.21: DiagnosticAging

Class	DiagnosticAuthRole			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to specify an authentication role that can be used to deliver fine-grained access rights. **Tags:** atp.recommendedPackage=DiagnosticAuthRoles			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bitPosition	PositiveInteger	0..1	attr	This attribute allows for the specification of the position of the enclosing role in a bitfield of roles.
isDefault	Boolean	0..1	attr	This attribute indicates whether the enclosing role is considered a default role.

Table A.22: DiagnosticAuthRole

Class	**DiagnosticAuthRoleProxy**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class indicates that an authentication is generally foreseen. The question whether the authentication is done in general or whether it is done role-specific depends on the existence of references to DiagAuthRole.			
Base	*ARObject*			
Aggregated by	DiagnosticAccessPermission.authenticationEnabled, DiagnosticMemoryDestinationUserDefined.authenticationEnabled			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication Role	DiagnosticAuthRole	*	ref	This reference identifies the authenticationRole applicable for the enclosing DiagnosticAccessPermission.

Table A.23: DiagnosticAuthRoleProxy

Class	***DiagnosticAuthentication*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::Authentication			
Note	This meta-class represents the ability to configure the usage of the UDS service Authentication in the Diagnostic extract.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticAuthTransmitCertificate, DiagnosticAuthenticationConfiguration, DiagnosticDeAuthentication, DiagnosticProofOfOwnership, DiagnosticVerifyCertificateBidirectional, DiagnosticVerifyCertificate Unidirectional			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication Class	Diagnostic AuthenticationClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference.

Table A.24: DiagnosticAuthentication

Class	**DiagnosticAuthenticationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service "authentication" on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.25: DiagnosticAuthenticationInterface

Class	**DiagnosticAuthenticationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the client authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Authentication	Diagnostic Authentication	0..1	ref	Reference to the DiagnosticAuthentication that is assigned to a SWC service port.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic AuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table A.26: DiagnosticAuthenticationPortMapping

Class	**DiagnosticClearCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.27: DiagnosticClearCondition

Class	**DiagnosticClearConditionPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticClearCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearCondition	DiagnosticClear Condition	0..1	ref	Reference to the ClearCondition which is mapped to a SWC service port.

▽

△

Class	**DiagnosticClearConditionPortMapping**			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ClearConditionMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.28: DiagnosticClearConditionPortMapping

Enumeration	**DiagnosticClearDtcLimitationEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	Scope of the DEM_ClearDTC Api.
Aggregated by	*DiagnosticMemoryDestination*.clearDtcLimitation
Literal	*Description*
allSupportedDtcs	DEM_ClearDtc API accepts all supported DTC values. **Tags:** atp.EnumerationLiteralIndex=0
clearAllDtcs	DEM_ClearDtc API accepts ClearAllDTCs only. **Tags:** atp.EnumerationLiteralIndex=1

Table A.29: DiagnosticClearDtcLimitationEnum

Enumeration	**DiagnosticClearEventAllowedBehaviorEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent
Note	This enumeration defines the possible behavior for clear event allowed
Aggregated by	DiagnosticEvent.clearEventAllowedBehavior
Literal	*Description*
noStatusByte Change	The event status byte keeps unchanged. **Tags:** atp.EnumerationLiteralIndex=0
onlyThisCycleAnd Readiness	The OperationCycle and readiness bits of the event status byte are reset. **Tags:** atp.EnumerationLiteralIndex=1

Table A.30: DiagnosticClearEventAllowedBehaviorEnum

Class	**DiagnosticComControl**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl			
Note	This represents an instance of the "Communication Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCommunicationControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**DiagnosticComControl**			
comControl Class	DiagnosticComControl Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticComControl in the given context.
customSub Function Number	PositiveInteger	0..1	attr	This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used.

Table A.31: DiagnosticComControl

Class	**DiagnosticComControlInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service communication control on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.32: DiagnosticComControlInterface

Class	«atpVariation» **DiagnosticCommonProps**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps			
Note	This meta-class aggregates a number of common properties that are shared among a diagnostic extract. **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Aggregated by	DiagnosticContributionSet.commonProperties			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
authentication Timeout	TimeValue	0..1	attr	This attribute defines the time (in seconds) that the authentication state is maintained in default-session if there is no communication from the authenticated client.
debounce AlgorithmProps	DiagnosticDebounce AlgorithmProps	*	aggr	Defines the used debounce algorithms relevant in the context of the enclosing DiagnosticCommonProps. Usually, there is a variety of debouncing algorithms to take into account and therefore the multiplicity of this aggregation is set to 0..*. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable; atpVariation **Tags:** vh.latestBindingTime=postBuild
default Endianness	ByteOrderEnum	0..1	attr	Defines the default endianness of the data belonging to a DID or RID which is applicable if the DiagnosticData Element does not define the endianness via the swData DefProps.baseType attribute.

▽

△

Class	«atpVariation» **DiagnosticCommonProps**			
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	"This aggregation represents the collection of diagnostic addresses that apply for the SoftwareClusterDesign. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable **Tags:** xml.namePlural=DIAGNOSTIC-ADDRESSES
event Combination Reporting Behavior	DiagnosticEvent CombinationReporting BehaviorEnum	0..1	attr	In case of EventCombination on Retrieval, this attribute specifies if a specific order of reporting is to be maintained.
external Authentication	DiagnosticExternal Authentication Identification	*	aggr	This aggregation supports the configuration of the authentication of diagnostic clients. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable
maxNumberOf Request Correctly Received Response Pending	PositiveInteger	0..1	attr	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceived-ResponsePending) allowed per request. DCM will send a negative response with response code 0x10 (generalReject), in case the limit value gets reached. Value 0xFF means that no limit number of NRC 0x78 response apply.
occurrence Counter Processing	DiagnosticOccurrence CounterProcessing Enum	0..1	attr	This attribute defines the consideration of the fault confirmation process for the occurrence counter.
resetConfirmed BitOnOverflow	Boolean	0..1	attr	This attribute defines, whether the confirmed bit is reset or not while an event memory entry will be displaced.
resetPendingBit OnOverflow	Boolean	0..1	attr	This attribute defines, whether the pending bit is reset or not while an event memory entry will be displaced. In order to be compliant to ISO 14229-1 [1], this parameter needs to be set to "false".
responseOnAll RequestSids	Boolean	0..1	attr	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).
responseOn Second Declined Request	Boolean	0..1	attr	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 BusyRepeat Request. FALSE: when the second request (Client B) can not be processed, it shall not be responded.
typeOfEvent Combination Supported	DiagnosticEvent CombinationBehavior Enum	0..1	attr	Select type of Event Combination support.

Table A.33: DiagnosticCommonProps

Class	***DiagnosticCondition*** (abstract)
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition
Note	Abstract element for StorageConditions and EnableConditions.
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*
Subclasses	DiagnosticClearCondition, DiagnosticEnableCondition, DiagnosticStorageCondition

▽

△

Class	***DiagnosticCondition*** (abstract)			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	Boolean	0..1	attr	Defines the initial status for enable or disable of acceptance/storage of event reports of a diagnostic event. The value is the initialization after power up (before this condition is reported the first time). true: acceptance/storage of a diagnostic event enabled false: acceptance/storage of a diagnostic event disabled

Table A.34: DiagnosticCondition

Class	**DiagnosticConditionInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.35: DiagnosticConditionInterface

Class	**DiagnosticConnectedIndicator**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	Description of indicators that are defined per DiagnosticEvent.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticEvent.connectedIndicator			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
behavior	DiagnosticConnected IndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycle	DiagnosticOperation Cycle	0..1	ref	The deactivation of indicators per event is defined as healing of a diagnostic event. The operation cycle in which the warning indicator will be switched off is defined here.
healingCycle Counter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
indicator	DiagnosticIndicator	0..1	ref	Reference to the used indicator.
indicatorFailure CycleCounter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of failure cycles for the WarningIndicatorOnCriteria. Please note that this attribute is not relevant for the Adaptive Platform.

Table A.36: DiagnosticConnectedIndicator

Class	DiagnosticContributionSet			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution			
Note	This meta-class represents a root node of a diagnostic extract. It bundles a given set of diagnostic model elements. The granularity of the DiagonsticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContribution Set. **Tags:** atp.recommendedPackage=DiagnosticContributionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
common Properties	DiagnosticCommon Props	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire Diagnostic ContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=commonProperties
element	DiagnosticCommon Element	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.diagnosticCommonElement, element.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceTable	DiagnosticServiceTable	*	ref	This represents the collection of DiagnosticServiceTables to be considered in the scope of this Diagnostic ContributionSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceTable.diagnosticServiceTable, service Table.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table A.37: DiagnosticContributionSet

Class	DiagnosticControlDTCSetting			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ControlDTCSetting			
Note	This represents an instance of the "Control DTC Setting" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticControlDtcSettings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dtcSettingClass	DiagnosticControlDTC SettingClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticControlDTCSetting in the given context.

Table A.38: DiagnosticControlDTCSetting

Class	DiagnosticCustomServiceInstance			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance			
Note	This meta-class has the ability to define an instance of a custom diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCustomInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
customService Class	DiagnosticCustom ServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustom ServiceClass.

Table A.39: DiagnosticCustomServiceInstance

Class	DiagnosticDTCInformationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.40: DiagnosticDTCInformationInterface

Class	***DiagnosticDataByIdentifier*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an abstract base class for all diagnostic services that access data by identifier.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticReadDataByIdentifier, DiagnosticReadScalingDataByIdentifier, DiagnosticWriteDataBy Identifier			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataIdentifier	DiagnosticAbstractData Identifier	0..1	ref	This represents the linked DiagnosticDataIdentifier.

Table A.41: DiagnosticDataByIdentifier

Class	DiagnosticDataElement			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	*ARObject*, *DiagnosticServiceMappingDiagTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*DiagnosticAbstractParameter*.dataElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**DiagnosticDataElement**			
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticData Element if used with DiagnosticReadScalingDataBy Identifier
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table A.42: DiagnosticDataElement

Class	**DiagnosticDataElementInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of an element of a diagnostic data identifier.

Table A.43: DiagnosticDataElementInterface

Class	**DiagnosticDataIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. **Tags:** atp.recommendedPackage=DiagnosticDataIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticAbstractDataIdentifier*, *DiagnosticCommon Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the Diagnostic DataIdentifier. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement.bitOffset, data Element.ident.shortName, dataElement.variation Point.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the Diagnostic DataIdentifier.

▽

△

Class	DiagnosticDataIdentifier			
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table A.44: DiagnosticDataIdentifier

Class	DiagnosticDataIdentifierGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.45: DiagnosticDataIdentifierGenericInterface

Class	DiagnosticDataIdentifierInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractDataIdentifierInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of a diagnostic data identifier.
write	ClientServerOperation	0..1	aggr	This represents the method to write the contents of a diagnostic data identifier.

Table A.46: DiagnosticDataIdentifierInterface

Class	DiagnosticDataIdentifierSet			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This represents the ability to define a list of DiagnosticDataIdentifiers that can be reused in different contexts. **Tags:** atp.recommendedPackage=DiagnosticDataIdentifierSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataIdentifier (ordered)	DiagnosticDataIdentifier	*	ref	Reference to an ordered list of Data Identifiers.

Table A.47: DiagnosticDataIdentifierSet

Class	DiagnosticDataPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticData Element	DiagnosticDataElement	0..1	ref	This reference represents the applicable DiagnosticData Element.
diagnosticData Identifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticData Identfiier.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype from which that data is obtained. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table A.48: DiagnosticDataPortMapping

Class	DiagnosticDebounceAlgorithmProps			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticDebouncingAlgorithm			
Note	Defines properties for the debounce algorithm class.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticCommonProps.debounceAlgorithmProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
debounce Algorithm	DiagEventDebounce Algorithm	0..1	aggr	This represents the actual debounce algorithm.

Table A.49: DiagnosticDebounceAlgorithmProps

Class	DiagnosticDoIPActivationLineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.50: DiagnosticDoIPActivationLineInterface

Class	**DiagnosticDoIPEntityIdentificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Entity Identification on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.51: DiagnosticDoIPEntityIdentificationInterface

Class	**DiagnosticDoIPGroupIdentificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.52: DiagnosticDoIPGroupIdentificationInterface

Class	**DiagnosticDoIPPowerModeInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.53: DiagnosticDoIPPowerModeInterface

Class	**DiagnosticDoIpActivationLinePortMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping::DoIpMapping
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP activation line. **Tags:** atp.recommendedPackage=DiagnosticPortMappings
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	DiagnosticDoIpActivationLinePortMapping			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.54: DiagnosticDoIpActivationLinePortMapping

Class	DiagnosticDoIpEntityIdentificationPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping::DoIpMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP identity identification. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.55: DiagnosticDoIpEntityIdentificationPortMapping

Class	DiagnosticDoIpGroupIdentificationPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping::DoIpMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP group identification. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.56: DiagnosticDoIpGroupIdentificationPortMapping

Class	DiagnosticDoIpPowerModePortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping::DoIpMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP power mode. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *AbstractDoIpPortMapping*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.57: DiagnosticDoIpPowerModePortMapping

Class	DiagnosticDownloadInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.58: DiagnosticDownloadInterface

Class	DiagnosticDynamicallyDefineDataIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
Note	This represents an instance of the "Dynamically Define Data Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataIdentifier	DiagnosticDynamicDataIdentifier	0..1	ref	This represents the applicable DiagnosticDynamicDataIdentfier.
dynamicallyDefineDataIdentifierClass	DiagnosticDynamicallyDefineDataIdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticDynamicallyDefineDataIdentifier in the given context.
maxSourceElement	PositiveInteger	0..1	attr	This represents the maximum number of source elements of the dynamically created DID.

Table A.59: DiagnosticDynamicallyDefineDataIdentifier

Class	DiagnosticDynamicallyDefineDataIdentifierClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
Note	This meta-class contains attributes shared by all instances of the "Dynamically Define Data Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
checkPerSourceId	Boolean	0..1	attr	If set to TRUE, the Dcm module shall check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0x F200 to 0xF3FF. If set to FALSE. the 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.

△

Class	**DiagnosticDynamicallyDefineDataIdentifierClass**			
configuration Handling	DiagnosticHandleDDDI ConfigurationEnum	0..1	attr	This configuration switch defines whether DDDID definition is handled as non-volatile information or not.
subfunction	DiagnosticDynamically DefineDataIdentifier SubfunctionEnum	*	attr	This attribute contains a list of applicable subfunctions for all DiagnosticDynamicallyDefineDataIdentifier that reference the DiagnosticDynamicallyDefineDataIdentifier Class.

Table A.60: DiagnosticDynamicallyDefineDataIdentifierClass

Class	**DiagnosticEcuReset**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
Note	This represents an instance of the "ECU Reset" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticEcuResets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
customSub Function Number	PositiveInteger	0..1	attr	This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used.
ecuResetClass	DiagnosticEcuReset Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticEcuReset in the given context.

Table A.61: DiagnosticEcuReset

Class	**DiagnosticEcuResetClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
Note	This meta-class contains attributes shared by all instances of the "Ecu Reset" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticEcuResets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
respondTo Reset	DiagnosticResponseTo EcuResetEnum	0..1	attr	This attribute defines whether the response to the Ecu Reset service shall be transmitted before or after the actual reset.

Table A.62: DiagnosticEcuResetClass

Class	**DiagnosticEcuResetInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service EcuReset on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces

▽

△

Class	DiagnosticEcuResetInterface			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.63: DiagnosticEcuResetInterface

Class	DiagnosticEnableCondition			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.64: DiagnosticEnableCondition

Class	DiagnosticEnableConditionPortMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticEnableCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
enableCondition	DiagnosticEnable Condition	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic EnableConditionPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.65: DiagnosticEnableConditionPortMapping

Class	***DiagnosticEnvCompareCondition*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	DiagnosticCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in DiagnosticCompareCondition.compareType.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Subclasses	DiagnosticEnvDataCondition, DiagnosticEnvDataElementCondition, DiagnosticEnvModeCondition			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareType	DiagnosticCompare TypeEnum	0..1	attr	This attributes represents the concrete type of the comparison.

Table A.66: DiagnosticEnvCompareCondition

Class	**DiagnosticEnvConditionFormula**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	A DiagnosticEnvConditionFormula embodies the computation instruction that is to be evaluated at runtime to determine if the DiagnosticEnvironmentalCondition is currently present (i.e. the formula is evaluated to true) or not (otherwise). The formula itself consists of parts which are combined by the logical operations specified by DiagnosticEnvConditionFormula.op. If a diagnostic functionality cannot be executed because an environmental condition fails then the diagnostic stack shall send a negative response code (NRC) back to the client. The value of the NRC is directly related to the specific formula and is therefore formalized in the attribute DiagnosticEnvCondition Formula.nrcValue.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part, DiagnosticEnvironmentalCondition.formula			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nrcValue	PositiveInteger	0..1	attr	This attribute represents the concrete NRC value that shall be returned if the condition fails.
op	DiagnosticLogical OperatorEnum	0..1	attr	This attribute represents the concrete operator (supported operators: and, or) of the condition formula.
part (ordered)	DiagnosticEnvCondition FormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

Table A.67: DiagnosticEnvConditionFormula

Class	***DiagnosticEnvConditionFormulaPart*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompare Condition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
Base	*ARObject*			
Subclasses	*DiagnosticEnvCompareCondition*, DiagnosticEnvConditionFormula			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.68: DiagnosticEnvConditionFormulaPart

Class	DiagnosticEnvDataCondition			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	A DiagnosticEnvDataCondition is an atomic condition that compares the current value of the referenced DiagnosticDataElement with a constant value defined by the ValueSpecification. All compareTypes are supported.			
Base	*ARObject*, *DiagnosticEnvCompareCondition*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compareValue	ValueSpecification	0..1	aggr	This attribute represents a fixed compare value taken to evaluate the compare condition.
dataElement	DiagnosticDataElement	0..1	ref	This reference represents the related diagnostic data element.

Table A.69: DiagnosticEnvDataCondition

Class	DiagnosticEnvironmentalCondition			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	The meta-class DiagnosticEnvironmentalCondition formalizes the idea of a condition which is evaluated during runtime of the ECU by looking at "environmental" states (e.g. one such condition is that the vehicle is not driving, i.e. vehicle speed == 0). **Tags:** atp.recommendedPackage=DiagnosticEnvironmentalConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
formula	DiagnosticEnvConditionFormula	0..1	aggr	This attribute represents the formula part of the DiagnosticEnvironmentalCondition.
modeElement	DiagnosticEnvModeElement	*	aggr	This aggregation contains a representation of Mode Declarations in the context of a DiagnosticEnvironmentalCondition.

Table A.70: DiagnosticEnvironmentalCondition

Class	DiagnosticEvent			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. **Tags:** atp.recommendedPackage=DiagnosticEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
associatedEventIdentification	PositiveInteger	0..1	attr	This attribute represents the identification number that is associated with the enclosing DiagnosticEvent and allows to identify it when placed into a snapshot record or extended data record storage. This value can be reported as internal data element in snapshot records or extended data records.
clearEventAllowedBehavior	DiagnosticClearEventAllowedBehaviorEnum	0..1	attr	This attribute defines the resulting UDS status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback

△

Class	**DiagnosticEvent**			
confirmation Threshold	PositiveInteger	0..1	attr	This attribute defines the number of operation cycles with a failed result before a confirmed DTC is set to 1. The semantic of this attribute is a by "1" increased value compared to the confirmation threshold of the "trip counter" mentioned in ISO 14229-1 in figure D.4. A value of "1" defines the immediate confirmation of the DTC along with the first reported failed. This is also sometimes called "zero trip DTC". A value of "2" defines a DTC confirmation in the operation cycle after the first occurred failed. A value of "2" is typically used in the US for OBD DTC confirmation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connectedIndicator.shortName, connected Indicator.variationPoint.shortLabel vh.latestBindingTime=postBuild
prestorage FreezeFrame	Boolean	0..1	attr	This attribute describes whether the Prestorage of Freeze Frames is supported by the assigned event or not. true: Prestorage of FreezeFrames is supported fFalse: Prestorage of FreezeFrames is not supported
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)
recoverableIn SameOperation Cycle	Boolean	0..1	attr	If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting PASSED will be ignored and not lead to a reset of the indication of a failed test.

Table A.71: DiagnosticEvent

Enumeration	**DiagnosticEventCombinationBehaviorEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
Note	Select type of Event Combination support
Aggregated by	DiagnosticCommonProps.typeOfEventCombinationSupported
Literal	***Description***
eventCombination OnRetrieval	Event combination on retrieval is used to combine events. For each event an individual event memory entry is created, while reporting the data via UDS, the data is combined. **Tags:** atp.EnumerationLiteralIndex=1
eventCombination OnStorage	Event combination on storage is used to combine events. Only one memory entry exists for each DTC which is also reported via UDS. **Tags:** atp.EnumerationLiteralIndex=0

Table A.72: DiagnosticEventCombinationBehaviorEnum

Enumeration	**DiagnosticEventCombinationReportingBehaviorEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
Note	Select reporting format of events. Applicable only for Event Combination on Retrieval.
Aggregated by	DiagnosticCommonProps.eventCombinationReportingBehavior
Literal	***Description***
reportingIn ChronlogicalOrder OldestFirst	The reporting order for event combination on retrieval is the chronological storage order of the events **Tags:** atp.EnumerationLiteralIndex=0

Table A.73: DiagnosticEventCombinationReportingBehaviorEnum

Enumeration	**DiagnosticEventDisplacementStrategyEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	Defines the displacement strategy.
Aggregated by	*DiagnosticMemoryDestination*.eventDisplacementStrategy
Literal	***Description***
full	Event memory entry displacement is enabled, by consideration of priority active/passive status, and occurrence. **Tags:** atp.EnumerationLiteralIndex=0
none	Event memory entry displacement is disabled. **Tags:** atp.EnumerationLiteralIndex=1
prioOcc	Event memory entry displacement is enabled, by consideration of priority and occurrence (but without active/passive status). **Tags:** atp.EnumerationLiteralIndex=2

Table A.74: DiagnosticEventDisplacementStrategyEnum

Class	**DiagnosticEventInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.75: DiagnosticEventInterface

Class	**DiagnosticEventPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticEvent is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	DiagnosticEventPortMapping			
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic EventPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.76: DiagnosticEventPortMapping

Class	DiagnosticEventToDebounceAlgorithmMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which Debounce Algorithm is applicable for a DiagnosticEvent. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
debounce Algorithm	DiagnosticDebounce AlgorithmProps	0..1	ref	Reference to a DebounceAlgorithm assigned to a DiagnosticEvent.
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which a Debounce Algorithm is assigned.

Table A.77: DiagnosticEventToDebounceAlgorithmMapping

Class	DiagnosticEventToEnableConditionGroupMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which EnableConditionGroup is applicable for a DiagnosticEvent. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which an Enable ConditionGroup is assigned.
enableCondition Group	DiagnosticEnable ConditionGroup	0..1	ref	Reference to an EnableConditionGroup assigned to a DiagnosticEvent.

Table A.78: DiagnosticEventToEnableConditionGroupMapping

Class	DiagnosticEventToOperationCycleMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which OperationCycle is applicable for a DiagnosticEvent. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which an Operation Cycle is assigned.
operationCycle	DiagnosticOperation Cycle	0..1	ref	Reference to an OperationCycle assigned to a Diagnostic Event.

Table A.79: DiagnosticEventToOperationCycleMapping

Class	DiagnosticEventToTroubleCodeUdsMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which UDS Diagnostic Trouble Code is applicable for a DiagnosticEvent. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which a UDS Diagnostic Trouble Code is assigned.
troubleCodeUds	DiagnosticTroubleCode Uds	0..1	ref	Reference to an UDS Diagnostic Trouble Code assigned to a DiagnosticEvent.

Table A.80: DiagnosticEventToTroubleCodeUdsMapping

Class	DiagnosticEventWindow			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
Note	This represents the ability to define the characteristics of the applicable event window			
Base	*ARObject*			
Aggregated by	DiagnosticResponseOnEvent.eventWindow			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventWindow Time	DiagnosticEventWindow TimeEnum	0..1	attr	This attribute clarifies the validity of the eventWindow

Table A.81: DiagnosticEventWindow

Class	DiagnosticExtendedDataRecord
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticExtendedDataRecord
Note	Description of an extended data record. **Tags:** atp.recommendedPackage=DiagnosticExtendedDataRecords
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticExtendedDataRecord**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
recordElement	DiagnosticParameter	*	aggr	Defined DataElements in the extended record element. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=recordElement.bitOffset, record Element.ident.shortName
recordNumber	PositiveInteger	0..1	attr	This attribute specifies an unique identifier for an extended data record.
trigger	DiagnosticRecord TriggerEnum	0..1	attr	This attribute specifies the primary trigger to allocate an event memory entry.
update	Boolean	0..1	attr	This attribute defines when an extended data record is captured. true: This extended data record is captured every time. false: This extended data record is only captured for new event memory entries.

Table A.82: DiagnosticExtendedDataRecord

Class	**DiagnosticExternalAuthenticationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic client authentication (i.e. convey the Authentication state to the Diagnostic Server instance of the DM) on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.83: DiagnosticExternalAuthenticationInterface

Class	**DiagnosticExternalAuthenticationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the external authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

▽

△

Class	DiagnosticExternalAuthenticationPortMapping			
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ClientAuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.84: DiagnosticExternalAuthenticationPortMapping

Class	DiagnosticFreezeFrame			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticFreezeFrame			
Note	This element describes combinations of DIDs for a non OBD relevant freeze frame. **Tags:** atp.recommendedPackage=DiagnosticFreezeFrames			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
recordNumber	PositiveInteger	0..1	attr	This attribute defines a record number for a freeze frame record. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
trigger	DiagnosticRecord TriggerEnum	0..1	attr	This attribute defines the primary trigger to allocate an event memory entry.
update	Boolean	0..1	attr	This attribute defines the approach when the freeze frame record is stored/updated. true: FreezeFrame record is captured every time. false: FreezeFrame record is only captured for new event memory entries.

Table A.85: DiagnosticFreezeFrame

Class	DiagnosticGenericUdsInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.86: DiagnosticGenericUdsInterface

Class	DiagnosticIndicatorInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.87: DiagnosticIndicatorInterface

Class	DiagnosticIndicatorPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticIndicator is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
indicator	DiagnosticIndicator	0..1	ref	Reference to the DiagnosticIndicator which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic IndicatorMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.88: DiagnosticIndicatorPortMapping

Class	***DiagnosticMapping*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Abstract element for different kinds of diagnostic mappings.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticEventToDebounceAlgorithmMapping, DiagnosticEventToEnableConditionGroupMapping, DiagnosticEventToOperationCycleMapping, DiagnosticEventToSecurityEventMapping, DiagnosticEventToTroubleCodeUdsMapping, DiagnosticFimAliasEventGroupMapping, DiagnosticFimAliasEventMapping, DiagnosticInhibitSourceEventMapping, DiagnosticMasterToSlaveEventMapping, DiagnosticProvidedDataMapping, DiagnosticSecureCodingMapping, DiagnosticSovdConfigurationDataIdentifierMapping, *DiagnosticSwMapping*, DiagnosticTroubleCodeUdsToClearConditionGroupMapping, DiagnosticTroubleCodeUdsToTroubleCodeObdMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.89: DiagnosticMapping

Class	***DiagnosticMemoryDestination*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
agingRequires TestedCycle	Boolean	0..1	attr	Defines whether the aging cycle counter is processed every aging cycles or else only tested aging cycle are considered. If the attribute is set to TRUE: only tested aging cycle are considered for aging cycle counter. If the attribute is set to FALSE: aging cycle counter is processed every aging cycle. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
clearDtc Limitation	DiagnosticClearDtc LimitationEnum	0..1	attr	Defines the scope of the DEM_ClearDTC Api. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
dtcStatus AvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.
event Displacement Strategy	DiagnosticEvent DisplacementStrategy Enum	0..1	attr	This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed.
maxNumberOf EventEntries	PositiveInteger	0..1	attr	This attribute fixes the maximum number of event entries in the fault memory.
memoryEntry StorageTrigger	DiagnosticMemoryEntry StorageTriggerEnum	0..1	attr	Describes the trigger to allocate an event memory entry.
statusBit HandlingTest FailedSinceLast Clear	DiagnosticStatusBit HandlingTestFailed SinceLastClearEnum	0..1	attr	This attribute defines, whether the aging and displacement mechanism shall be applied to the "Test FailedSinceLastClear" status bits. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
typeOfFreeze FrameRecord Numeration	DiagnosticTypeOf FreezeFrameRecord NumerationEnum	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

Table A.90: DiagnosticMemoryDestination

Class	**DiagnosticMemoryDestinationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticMemoryDestination. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Destination	DiagnosticMemory Destination	0..1	ref	Reference to the MemoryDestination which is mapped to a SWC service port.

△

Class	DiagnosticMemoryDestinationPortMapping			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MemoryDestinationMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.91: DiagnosticMemoryDestinationPortMapping

Class	DiagnosticMemoryDestinationPrimary			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
Note	This represents a primary memory for a diagnostic event. **Tags:** atp.recommendedPackage=DiagnosticMemoryDestinations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory Destination*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
typeOfDtc Supported	DiagnosticTypeOfDtc SupportedEnum	0..1	attr	This attribute defines the format returned by Dem_Dcm GetTranslationType and does not relate to/influence the supported Dem functionality.

Table A.92: DiagnosticMemoryDestinationPrimary

Class	DiagnosticMemoryDestinationUserDefined			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
Note	This represents a user-defined memory for a diagnostic event. **Tags:** atp.recommendedPackage=DiagnosticMemoryDestinations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory Destination*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
affectedBy Control DTCSetting	Boolean	0..1	attr	This attribute configures how the content of the memory is affected by an active ControlDTCSetting or not: • If the attribute is set to **true**, the user-defined fault memory is **not** updated if ControlDTCSetting is off. • If the attribute is set to **false**, the user defined fault memory is updated even if ControlDTCSetting is off.
authentication Enabled	DiagnosticAuthRole Proxy	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=authenticationEnabled
memoryId	PositiveInteger	0..1	attr	This represents the identifier of the user-defined memory.

Table A.93: DiagnosticMemoryDestinationUserDefined

Class	DiagnosticMonitorInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.94: DiagnosticMonitorInterface

Class	DiagnosticMonitorPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port the Diagnostic Monitor is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.95: DiagnosticMonitorPortMapping

Class	DiagnosticMultipleConditionInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a condition-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticConditionInterface, the DiagnosticMultipleCondition Interface allows for handling more than one condition in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.96: DiagnosticMultipleConditionInterface

Class	**DiagnosticMultipleConditionPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port that can handle a collection of diagostic conditions the specific condition is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Condition	DiagnosticCondition	0..1	ref	Reference to the DiagnosticCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ConditionPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.97: DiagnosticMultipleConditionPortMapping

Class	**DiagnosticMultipleEventInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a event-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticEventInterface, the DiagnosticMultipleMonitorInterface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.98: DiagnosticMultipleEventInterface

Class	**DiagnosticMultipleEventPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port that can handle a collection of event status the specific event is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	DiagnosticMultipleEventPortMapping			
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleEventPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.99: DiagnosticMultipleEventPortMapping

Class	DiagnosticMultipleMonitorInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticMonitorInterface, the DiagnosticMultipleMonitor Interface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.100: DiagnosticMultipleMonitorInterface

Class	DiagnosticMultipleMonitorPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port that can handle a collection of monitors the specific event is mapped **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

▽

△

Class	DiagnosticMultipleMonitorPortMapping			
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleMonitorPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.101: DiagnosticMultipleMonitorPortMapping

Class	*DiagnosticMultipleResourcePortMapping* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This abstract base class enables the mapping of diagnostic PortInterfaces that deal with multiple diagnostic resources.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMultipleConditionPortMapping, DiagnosticMultipleEventPortMapping, DiagnosticMultiple MonitorPortMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
overrideId	PositiveInteger	0..1	attr	This attribute shall be used to define the value of a manually override of the automatic generated handle Id value.

Table A.102: DiagnosticMultipleResourcePortMapping

Enumeration	DiagnosticOccurrenceCounterProcessingEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
Note	The occurrence counter triggering types.
Aggregated by	DiagnosticCommonProps.occurrenceCounterProcessing
Literal	**Description**
confirmedDtcBit	The occurrence counter is incremented when TestFailed bit transitions from 0 to 1 if the fault confirmation was successful (ConfirmedDTC bit is already set). **Tags:** atp.EnumerationLiteralIndex=0
testFailedBit	The occurrence counter is incremented when TestFailed bit transitions from 0 to 1 (and the fault confirmation is not considered). **Tags:** atp.EnumerationLiteralIndex=1

Table A.103: DiagnosticOccurrenceCounterProcessingEnum

Class	DiagnosticOperationCycle			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle			
Note	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. **Tags:** atp.recommendedPackage=DiagnosticOperationCycles			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	DiagnosticOperation CycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table A.104: DiagnosticOperationCycle

Class	DiagnosticOperationCycleInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.105: DiagnosticOperationCycleInterface

Class	DiagnosticOperationCyclePortMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticOperationCycle is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
operationCycle	DiagnosticOperationCycle	0..1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototypeInExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticOperationCyclePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef

Table A.106: DiagnosticOperationCyclePortMapping

Class	DiagnosticParameter			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe information relevant for the execution of a specific diagnostic service, i.e. it can be taken to parameterize the service.			
Base	*ARObject*, *DiagnosticAbstractParameter*			
Aggregated by	DiagnosticDataIdentifier.dataElement, DiagnosticExtendedDataRecord.recordElement, DiagnosticInfoType.dataElement, DiagnosticParameterIdentifier.dataElement, DiagnosticRequestRoutineResults.request, DiagnosticRequestRoutineResults.response, DiagnosticStartRoutine.request, DiagnosticStartRoutine.response, DiagnosticStopRoutine.request, DiagnosticStopRoutine.response			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**DiagnosticParameter**			
ident	DiagnosticParameter Ident	0..1	aggr	The aggregation in the role ident provides the ability to make the DiagnosticAbstractParameter identifiable. From the semantical point of view, the AbstractDiagnostic Parameter is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let AbstractDiagnostic Parameter directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor
supportInfo	DiagnosticParameter SupportInfo	0..1	aggr	This attribute represents the ability to define which bit of the support info byte is representing this part of the PID.

Table A.107: DiagnosticParameter

Class	**DiagnosticPeriodicRate**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
Note	This represents the ability to define a periodic rate for the specification of the "read data by periodic ID" diagnostic service.			
Base	*ARObject*			
Aggregated by	DiagnosticReadDataByPeriodicIDClass.periodicRate			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
period	TimeValue	0..1	attr	This represents the period of the DiagnosticPeriodicRate in seconds.
periodicRate Category	DiagnosticPeriodicRate CategoryEnum	0..1	attr	This attribute represents the category of the periodic rate.

Table A.108: DiagnosticPeriodicRate

Class	***DiagnosticPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticAuthentication Interface, DiagnosticComControlInterface, DiagnosticConditionInterface, DiagnosticDTCInformation Interface, DiagnosticDoIPActivationLineInterface, DiagnosticDoIPEntityIdentificationInterface, Diagnostic DoIPGroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDoIPTriggerVehicle AnnouncementInterface, DiagnosticDownloadInterface, DiagnosticEcuResetInterface, DiagnosticEvent Interface, DiagnosticExternalAuthenticationInterface, DiagnosticGenericUdsInterface, Diagnostic IndicatorInterface, DiagnosticMonitorInterface, *DiagnosticMultipleResourceInterface*, Diagnostic OperationCycleInterface, DiagnosticRequestFileTransferInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidationInterface, *DiagnosticSovdPortInterface*, DiagnosticTransmitCertificate Interface, DiagnosticUploadInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.109: DiagnosticPortInterface

Class	**DiagnosticProvidedDataMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
Note	This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly. **Tags:** atp.recommendedPackage=DataMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider.
dataProvider	NameToken	0..1	attr	This represents the ability to further specify the data provider.

Table A.110: DiagnosticProvidedDataMapping

Class	**DiagnosticReadDTCInformation**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDTCInformation			
Note	This represents an instance of the "Read DTC Information" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticReadDtcInformations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
read DTCInformation Class	DiagnosticReadDTC InformationClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRead DTCInformation in the given context.

Table A.111: DiagnosticReadDTCInformation

Class	**DiagnosticReadDataByIdentifier**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an instance of the "Read Data by Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDataByIdentifier*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
readClass	DiagnosticReadDataBy IdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticReadDataBy Identifier in the given context.

Table A.112: DiagnosticReadDataByIdentifier

Class	**DiagnosticReadDataByIdentifierClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This meta-class contains attributes shared by all instances of the "Read Data by Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
maxDidToRead	PositiveInteger	0..1	attr	This attribute represents the maximum number of allowed DIDs in a single instance of DiagnosticReadDataBy Identifier.

Table A.113: DiagnosticReadDataByIdentifierClass

Class	**DiagnosticReadDataByPeriodicIDClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
Note	This meta-class contains attributes shared by all instances of the "Read Data by periodic Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticReadDataByPeriodicIds			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
maxPeriodicDid ToRead	PositiveInteger	0..1	attr	This represents the maximum number of data identifiers that can be included in one request.
periodicRate	DiagnosticPeriodicRate	*	aggr	This represents the description of a collection of periodic rates in which the service can be executed.
schedulerMax Number	PositiveInteger	0..1	attr	This represents the maximum number of periodic data identifiers that can be scheduled in parallel.

Table A.114: DiagnosticReadDataByPeriodicIDClass

Enumeration	**DiagnosticRecordTriggerEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticFreezeFrame
Note	Triggers to allocate an event memory entry.
Aggregated by	DiagnosticExtendedDataRecord.trigger, DiagnosticFreezeFrame.trigger
Literal	*Description*
confirmed	capture on "Confirmed" **Tags:** atp.EnumerationLiteralIndex=0
custom	implement custom capture **Tags:** atp.EnumerationLiteralIndex=4
fdcThreshold	capture on "FDC Threshold" **Tags:** atp.EnumerationLiteralIndex=1
pending	capture on "Pending" **Tags:** atp.EnumerationLiteralIndex=2
testFailed	capture on "Test Failed" **Tags:** atp.EnumerationLiteralIndex=3

Table A.115: DiagnosticRecordTriggerEnum

Class	**DiagnosticRequestFileTransferInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for file transfer using diagnostic channels on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.116: DiagnosticRequestFileTransferInterface

Class	**DiagnosticRequestRoutineResults**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to define the result of a diagnostic routine execution.			
Base	*ARObject*, *DiagnosticRoutineSubfunction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticRoutine.requestResult			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
request	DiagnosticParameter	*	aggr	This represents the request parameters.
response	DiagnosticParameter	*	aggr	This represents the response parameters.

Table A.117: DiagnosticRequestRoutineResults

Class	**DiagnosticResponseOnEvent**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
Note	This represents an instance of the "Response on Event" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticResponseOnEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventWindow	DiagnosticEventWindow	*	aggr	This represents the applicable DiagnosticEventWindows
responseOn EventAction	DiagnosticResponseOn EventActionEnum	0..1	attr	Defines sub-functions of the service ResponseOnEvent.
responseOn EventClass	DiagnosticResponseOn EventClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticResponseOnEvent in the given context.

Table A.118: DiagnosticResponseOnEvent

Enumeration	**DiagnosticResponseOnEventActionEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent
Note	This meta-class has the ability to define sub-functions of the UDS service ResponseOnEvent.
Aggregated by	DiagnosticResponseOnEvent.responseOnEventAction
Literal	***Description***

▽

△

Enumeration	**DiagnosticResponseOnEventActionEnum**
clear	Clears the configured events. **Tags:** atp.EnumerationLiteralIndex=2
onChangeOfData Identifier	Reports based on change of data identifier. **Tags:** atp.EnumerationLiteralIndex=6
onComparisonOf Values	Triggered if data condition is met (e.g. RPM over 5000 1/min). **Tags:** atp.EnumerationLiteralIndex=8
onDTCStatus Change	Reports based on change of DTC status. **Tags:** atp.EnumerationLiteralIndex=7
report	Reports the activated events. **Tags:** atp.EnumerationLiteralIndex=3
reportDTCRecord InformationOnDtc StatusChange	Reports the DTC record-related information based on a DTC status change. (Subfunction 0x09) **Tags:** atp.EnumerationLiteralIndex=5
reportMostRecent DtcOnStatus Change	Triggers the report of the most recent failed or confirmed DTC (Subfunction 0x08). **Tags:** atp.EnumerationLiteralIndex=4
start	Starts the response on event service. **Tags:** atp.EnumerationLiteralIndex=1
stop	Stops the response on event service. **Tags:** atp.EnumerationLiteralIndex=0

Table A.119: DiagnosticResponseOnEventActionEnum

Class	**DiagnosticResponseOnEventClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
Note	This represents the ability to define common properties for all instances of the "Response on Event" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticResponseOnEvents			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxNumberOf Stored DTCStatus ChangedEvents	PositiveInteger	0..1	attr	The maximum number of DTCs that can be stored as DTCs with change status within one ResponseOnEvent SchedulerRate interval.
maxNum ChangeOfData IdentfierEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function onChangeOf DataIdentifier.
maxNum ComparisionOf ValueEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function on ComparisonOfValues.
maxSupported DIDLength	PositiveInteger	0..1	attr	The maximum number of measurable data bytes allowed for each DID that is used for comparison or data change.
responseOn EventScheduler Rate	TimeValue	0..1	attr	The call rate of the periodic scheduler to compare the values of the DataIdentifier (DID) or to detect DTC status changes.

▽

△

Class	**DiagnosticResponseOnEventClass**			
storeEvent Enabled	Boolean	0..1	attr	Specifies if the storeEvent functionality of the Response OnEvent diagnostic service shall be supported or not. If set to true, the storeEvent functionality is available. If set to false the storeEvent functionality is not available.

Table A.120: DiagnosticResponseOnEventClass

Enumeration	**DiagnosticResponseToEcuResetEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset
Note	This enumeration controls the point in time in which a response to the reception of an EcuReset service shall be generated.
Aggregated by	DiagnosticEcuResetClass.respondToReset
Literal	*Description*
respondAfterReset	Answer to EcuReset service should come after the reset. **Tags:** atp.EnumerationLiteralIndex=0
respondBefore Reset	Answer to EcuReset service should come before the reset. **Tags:** atp.EnumerationLiteralIndex=1

Table A.121: DiagnosticResponseToEcuResetEnum

Class	**DiagnosticRoutine**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to define a diagnostic routine. **Tags:** atp.recommendedPackage=DiagnosticRoutines			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticRoutine in the scope of diagnostic workflow **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
requestResult	DiagnosticRequest RoutineResults	0..1	aggr	This represents the ability to request the result of a running routine.
routineInfo	PositiveInteger	0..1	attr	This represents the routine info byte. The info byte contains a manufacturer-specific value (for the identification of record identifiers) that is reported to the tester. Other use cases for this attribute are mentioned in ISO 27145 and ISO 26021.
start	DiagnosticStartRoutine	0..1	aggr	This represents the ability to start a routine
stop	DiagnosticStopRoutine	0..1	aggr	This represents the ability to stop a running routine.

Table A.122: DiagnosticRoutine

Class	DiagnosticRoutineGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.123: DiagnosticRoutineGenericInterface

Class	DiagnosticRoutineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticAbstractRoutineInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requestResult	ClientServerOperation	0..1	aggr	This represents the request result method of the diagnostic routine.
start	ClientServerOperation	0..1	aggr	This represents the start method of the diagnostic routine.
stop	ClientServerOperation	0..1	aggr	This represents the stop method of the diagnostic routine.

Table A.124: DiagnosticRoutineInterface

Class	***DiagnosticRoutineSubfunction*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class acts as an abstract base class to routine subfunctions.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DiagnosticRequestRoutineResults, DiagnosticStartRoutine, DiagnosticStopRoutine			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
access Permission	DiagnosticAccess Permission	0..1	ref	This reference represents the access permission of the owning routine subfunction.

Table A.125: DiagnosticRoutineSubfunction

Class	DiagnosticSecurityAccess
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess
Note	This represents an instance of the "Security Access" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticSecurityAccesss
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticSecurityAccess**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requestSeedId	PositiveInteger	0..1	attr	This would be 0x01, 0x03, 0x05, ... The sendKey id can be computed by adding 1 to the requestSeedId
securityAccessClass	DiagnosticSecurityAccessClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticSecurityAccess in the given context.
securityDelayTimeOnBoot	TimeValue	0..1	attr	Start delay timer on power on in seconds. This delay indicates the time after ECU boot power-on where no security access request is accepted.
securityLevel	DiagnosticSecurityLevel	0..1	ref	This reference identifies the applicable security level for the security access. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=securityLevel

Table A.126: DiagnosticSecurityAccess

Class	**DiagnosticSecurityAccessClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticSecurityAccesss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sharedTimer	Boolean	0..1	attr	Switch between separate or single shared timer instance and timer value. • true: use shared timer instance and timer value for all security access levels combined. • false: use separate timer instance and timer values for each security level.

Table A.127: DiagnosticSecurityAccessClass

Class	**DiagnosticSecurityLevel**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a security level considered for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticSecurityLevels			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessDataRecordSize	PositiveInteger	0..1	attr	This represents the size of the AccessDataRecord used in GetSeed. Unit:byte.
keySize	PositiveInteger	0..1	attr	This represents the size of the security key. Unit: byte.

▽

△

Class	DiagnosticSecurityLevel			
numFailed SecurityAccess	PositiveInteger	0..1	attr	This represents the number of failed security accesses after which the delay time is activated.
securityDelay Time	TimeValue	0..1	attr	This represents the delay time after a failed security access. Unit: second.
seedSize	PositiveInteger	0..1	attr	This represents the size of the security seed. Unit: byte.

Table A.128: DiagnosticSecurityLevel

Class	DiagnosticSecurityLevelInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.129: DiagnosticSecurityLevelInterface

Class	DiagnosticSecurityLevelPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticSecurityLevel is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SecurityLevelMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
securityLevel	DiagnosticSecurityLevel	0..1	ref	Reference to the SecurityLevel which is mapped to a SWC service port.

Table A.130: DiagnosticSecurityLevelPortMapping

Class	***DiagnosticServiceClass*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This meta-class provides the ability to define common properties that are shared among all instances of sub-classes of DiagnosticServiceInstance.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticAuthenticationClass, DiagnosticClearDiagnosticInformationClass, DiagnosticClearReset EmissionRelatedInfoClass, DiagnosticComControlClass, DiagnosticControlDTCSettingClass, Diagnostic CustomServiceClass, DiagnosticDataTransferClass, DiagnosticDynamicallyDefineDataIdentifierClass, DiagnosticEcuResetClass, DiagnosticIoControlClass, DiagnosticReadDTCInformationClass, Diagnostic ReadDataByIdentifierClass, DiagnosticReadDataByPeriodicIDClass, DiagnosticReadMemoryByAddress Class, DiagnosticReadScalingDataByIdentifierClass, DiagnosticRequestControlOfOnBoardDeviceClass, DiagnosticRequestCurrentPowertrainDataClass, DiagnosticRequestDownloadClass, DiagnosticRequest EmissionRelatedDTCClass, DiagnosticRequestEmissionRelatedDTCPermanentStatusClass, Diagnostic RequestFileTransferClass, DiagnosticRequestOnBoardMonitoringTestResultsClass, DiagnosticRequest PowertrainFreezeFrameDataClass, DiagnosticRequestUploadClass, DiagnosticRequestVehicleInfo Class, DiagnosticResponseOnEventClass, DiagnosticRoutineControlClass, DiagnosticSecurityAccess Class, DiagnosticSessionControlClass, DiagnosticTransferExitClass, DiagnosticWriteDataByIdentifier Class, DiagnosticWriteMemoryByAddressClass			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.131: DiagnosticServiceClass

Class	**DiagnosticServiceGenericMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic ServiceInstance	DiagnosticService Instance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ServiceGenericMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table A.132: DiagnosticServiceGenericMapping

Class	***DiagnosticServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents a concrete instance of a diagnostic service.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*DiagnosticAuthentication*, DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelatedInfo, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, *DiagnosticDataByIdentifier*, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, DiagnosticIOControl, *DiagnosticMemoryByAddress*, DiagnosticReadDTCInformation, DiagnosticReadDataByPeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequestPowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, DiagnosticRoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
access Permission	DiagnosticAccessPermission	0..1	ref	This represents the collection of DiagnosticAccessPermissions that allow for the execution of the referencing DiagnosticServiceInstance.. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=accessPermission
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table A.133: DiagnosticServiceInstance

Class	**DiagnosticServiceValidationConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class has the ability to configure the order of manufacturer/supplier-checks. **Tags:** atp.recommendedPackage=DiagnosticValueConfigurations			
Base	*ARObject*			
Aggregated by	SoftwareClusterDiagnosticDeploymentProps.validationConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
manufacturer ValidationOrder (ordered)	DiagnosticServiceValidationMapping	*	ref	This reference defines the order in which validations created by manufacturer are executed.
sovdValidation Order (ordered)	DiagnosticSovdServiceValidationPortMapping	*	ref	This reference defines the order in which validations of SOVD requests are executed. **Tags:** atp.Status=candidate
supplier ValidationOrder (ordered)	DiagnosticServiceValidationMapping	*	ref	This reference defines the order in which validations created by supplier are executed.

Table A.134: DiagnosticServiceValidationConfiguration

Class	DiagnosticServiceValidationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.135: DiagnosticServiceValidationInterface

Class	DiagnosticServiceValidationMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to specify manufacturer/supplier checks to be executed before diagnostic services can be processed. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This mapping identifies a PortPrototype typed by a DiagnosticValidationInterface in which a manufacturer/ supplier-specific check is executed. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table A.136: DiagnosticServiceValidationMapping

Class	DiagnosticSession			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a diagnostic session. **Tags:** atp.recommendedPackage=DiagnosticSessions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticSession in the scope of diagnostic workflow
p2ServerMax	TimeValue	0..1	attr	This is the session value for P2ServerMax in seconds (per Session Control). The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds.

△

Class	DiagnosticSession			
p2StarServer Max	TimeValue	0..1	attr	This is the session value for P2*ServerMax in seconds (per Session Control). The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds.

Table A.137: DiagnosticSession

Class	DiagnosticSessionControlClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SessionControl			
Note	This meta-class contains attributes shared by all instances of the "Session Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticSessionControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
s3Server Timeout	TimeValue	0..1	attr	Time for the server to keep a diagnostic session other than the default session active while not receiving any diagnostic request message.

Table A.138: DiagnosticSessionControlClass

Enumeration	DiagnosticSignificanceEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	Significance level of a diagnostic event.
Aggregated by	DiagnosticTroubleCodeProps.significance
Literal	**Description**
fault	Failure, which affects the component/ECU itself. **Tags:** atp.EnumerationLiteralIndex=0
occurence	Issue, which indicates additional information concerning insufficient system behavior. **Tags:** atp.EnumerationLiteralIndex=1

Table A.139: DiagnosticSignificanceEnum

Class	DiagnosticSovdAuthorizationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD authorization. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.140: DiagnosticSovdAuthorizationInterface

Class	DiagnosticSovdAuthorizationPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD authorization. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdAuthorizationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate

Table A.141: DiagnosticSovdAuthorizationPortMapping

Class	DiagnosticSovdBulkDataInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD bulk data transmission. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.142: DiagnosticSovdBulkDataInterface

Class	DiagnosticSovdBulkDataPortMapping
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdBulkDataInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdBulkData. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticSovdBulkDataPortMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdBulkDataPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovdBulk Data	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table A.143: DiagnosticSovdBulkDataPortMapping

Class	**DiagnosticSovdConfigurationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to configure a PortInterface for the exchange of configuration content. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.144: DiagnosticSovdConfigurationInterface

Class	**DiagnosticSovdConfigurationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdConfigurationInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdConfiguration. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticSovdConfigurationPortMapping**			
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdConfigurationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovd Configuration	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table A.145: DiagnosticSovdConfigurationPortMapping

Class	**DiagnosticSovdFaultMemoryAccess**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents the ability to access (read/get and delete) the fault memory. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.146: DiagnosticSovdFaultMemoryAccess

Class	**DiagnosticSovdLog**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Log" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.147: DiagnosticSovdLog

Class	**DiagnosticSovdMethod**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	A DiagnosticSovdMethod represents a re-usable complex operation (that consists of primitive operations) in the context of the communication of an SOVD server. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdMethods			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
delete	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Status=candidate
get	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "get" method primitive. **Tags:** atp.Status=candidate
post	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "post" method primitive. **Tags:** atp.Status=candidate
put	DiagnosticSovdMethodPrimitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Status=candidate

Table A.148: DiagnosticSovdMethod

Class	**DiagnosticSovdProximityChallengeInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD proximity challenge. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.149: DiagnosticSovdProximityChallengeInterface

Class	**DiagnosticSovdProximityChallengePortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD proximity challenge. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticSovdProximityChallengePortMapping**			
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdProximityChallengePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate

Table A.150: DiagnosticSovdProximityChallengePortMapping

Class	**DiagnosticSovdServiceValidationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD service validation. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterface			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.151: DiagnosticSovdServiceValidationInterface

Class	**DiagnosticSovdServiceValidationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD service validation. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdValidationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef

▽

△

Class	DiagnosticSovdServiceValidationPortMapping			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Tags:** atp.Status=candidate

Table A.152: DiagnosticSovdServiceValidationPortMapping

Class	DiagnosticSovdUpdateInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD update procedure. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.153: DiagnosticSovdUpdateInterface

Class	DiagnosticSovdUpdatePortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping associates a PPortPrototype typed by an DiagnosticSovdUpdateInterface with the corresponding SOVD service instance that is modeled as a DiagnosticSovdUpdate. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdUpdatePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovdUpdate	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table A.154: DiagnosticSovdUpdatePortMapping

Class	**DiagnosticStartRoutine**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This represents the ability to start a diagnostic routine.			
Base	*ARObject*, *DiagnosticRoutineSubfunction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticRoutine.start			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
request	DiagnosticParameter	*	aggr	This represents the request parameters.
response	DiagnosticParameter	*	aggr	This represents the response parameters.

Table A.155: DiagnosticStartRoutine

Enumeration	**DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.
Aggregated by	*DiagnosticMemoryDestination*.statusBitHandlingTestFailedSinceLastClear
Literal	*Description*
statusBitAgingAnd Displacement	The "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies. **Tags:** atp.EnumerationLiteralIndex=0
statusBitNormal	Aging and displacement has no impact on the "TestFailedSinceLastClear" status bits. **Tags:** atp.EnumerationLiteralIndex=1

Table A.156: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum

Class	**DiagnosticStopRoutine**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This represents the ability to stop a diagnostic routine.			
Base	*ARObject*, *DiagnosticRoutineSubfunction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticRoutine.stop			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
request	DiagnosticParameter	*	aggr	This represents the request parameters.
response	DiagnosticParameter	*	aggr	This represents the response parameters.

Table A.157: DiagnosticStopRoutine

Class	**DiagnosticTransmitCertificateInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement the transmit-certificate functionality on application software level. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table A.158: DiagnosticTransmitCertificateInterface

Class	**DiagnosticTroubleCodeGroup**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	The diagnostic trouble code group defines the DTCs belonging together and thereby forming a group. **Tags:** atp.recommendedPackage=DiagnosticTroubleCodes			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dtc	DiagnosticTroubleCode	*	ref	This represents the collection of DiagnosticTroubleCodes defined by this DiagnosticTroubleCodeGroup. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dtc.diagnosticTroubleCode, dtc.variation Point.shortLabel vh.latestBindingTime=postBuild
groupNumber	PositiveInteger	0..1	attr	This represents the base number of the DTC group. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.159: DiagnosticTroubleCodeGroup

Class	**DiagnosticTroubleCodeProps**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This element defines common Dtc properties that can be reused by different DTCs. **Tags:** atp.recommendedPackage=DiagnosticTroubleCodePropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aging	DiagnosticAging	0..1	ref	Reference to an aging algorithm in case that an aging/ unlearning of the event is allowed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=aging
diagnostic Memory	DiagnosticMemory Destination	0..1	ref	Reference to the applicable DiagnosticMemory Destination. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticMemory
extendedData Record	DiagnosticExtended DataRecord	*	ref	Defines the links to an extended data class sampler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=extendedDataRecord.diagnosticExtended DataRecord, extendedDataRecord.variationPoint.short Label vh.latestBindingTime=preCompileTime
freezeFrame	DiagnosticFreezeFrame	*	ref	Define the links to a freeze frame class sampler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=freezeFrame.diagnosticFreezeFrame, freeze Frame.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

△

Class	**DiagnosticTroubleCodeProps**			
immediateNv DataStorage	Boolean	0..1	attr	Change description for Class immediateNvDataStorage in table "Table A.111: DiagnosticTroubleCodeProps": Switch to enable immediate storage triggering of an according event memory entry persistently to NVRAM. true: immediate non-volatile storage triggering on first occurrence and shutdown. false: immediate non-volatile storage triggering on shutdown.
legislated FreezeFrame ContentUdsObd	DiagnosticDataIdentifier Set	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the UDS protocol such as OBDonUDS or WWH-OBD. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=legislatedFreezeFrameContentUds Obd.diagnosticDataIdentifierSet, legislatedFreezeFrame ContentUdsObd.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
maxNumber FreezeFrame Records	PositiveInteger	0..1	attr	This attribute defines the number of according freeze frame records, which can maximal be stored for this event. Therefore all these freeze frame records have the same freeze frame class.
obdProps	DiagnosticTroubleCode ObdProps	0..1	aggr	This aggregation is used to define OBD-relevant properties for a Diagnostic Trouble Code **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=obdProps, obdProps.variationPoint.short Label vh.latestBindingTime=postBuild
priority	PositiveInteger	0..1	attr	Priority of the event, in view of full event buffer. A lower value means higher priority. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
significance	DiagnosticSignificance Enum	0..1	attr	Significance of the event, which indicates additional information concerning fault classification and resolution.
snapshot RecordContent	DiagnosticDataIdentifier Set	0..1	ref	This represents the freeze frame layout as a set of DIDs. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=snapshotRecordContent.diagnosticData IdentifierSet, snapshotRecordContent.variationPoint.short Label vh.latestBindingTime=preCompileTime

Table A.160: DiagnosticTroubleCodeProps

Class	**DiagnosticTroubleCodeUds**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This element is used to describe diagnostic trouble codes (DTCs). **Tags:** atp.recommendedPackage=DiagnosticTroubleCodes			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticTroubleCode*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticTroubleCodeUds**			
considerPto Status	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. true: the event is affected by the Dem PTO handling. false: the event is not affected by the Dem PTO handling. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
eventReadiness Group	EventObdReadiness Group	0..1	aggr	This attribute specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This attribute is only applicable for emission-related ECUs. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=eventReadinessGroup.eventObdReadiness Group, eventReadinessGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild
functionalUnit	PositiveInteger	0..1	attr	This attribute specifies a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity information. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
obdDtc Value3Byte	PositiveInteger	0..1	attr	3 Byte OBD DTC value based on the definition from SAE J2012. The existence of this attribute is only required if separated UDS and OBD DTC values are used for SAE J1979-2. If this attribute does not exist, then UDS DTC values are used with J1979-2. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
severity	DiagnosticUdsSeverity Enum	0..1	attr	DTC severity according to ISO 14229-1. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
troubleCode Props	DiagnosticTroubleCode Props	0..1	ref	Defined properties associated with the DemDTC. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=troubleCodeProps.diagnosticTroubleCode Props, troubleCodeProps.variationPoint.shortLabel vh.latestBindingTime=postBuild
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
wwhObdDtc Class	DiagnosticWwhObdDtc ClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.161: DiagnosticTroubleCodeUds

Class	**DiagnosticTroubleCodeUdsToClearConditionGroupMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. **Tags:** atp.recommendedPackage=DiagnosticMappings

▽

△

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
Base	*AUElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearCondition Group	DiagnosticClear ConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClear ConditionGroup.
troubleCodeUds	DiagnosticTroubleCode Uds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping.

Table A.162: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

Enumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	FreezeFrame record numeration type
Aggregated by	*DiagnosticMemoryDestination*.typeOfFreezeFrameRecordNumeration
Literal	**Description**
calculated	Freeze frame records will be numbered consecutive starting by 1 in their chronological order. **Tags:** atp.EnumerationLiteralIndex=0
configured	Freeze frame records will be numbered based on the given configuration in their chronological order. **Tags:** atp.EnumerationLiteralIndex=1

Table A.163: DiagnosticTypeOfFreezeFrameRecordNumerationEnum

Class	DiagnosticUploadInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*AUElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.164: DiagnosticUploadInterface

Class	DiagnosticWriteDataByIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an instance of the "Write Data by Identifier" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	*AUElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticDataByIdentifier*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**DiagnosticWriteDataByIdentifier**			
writeClass	DiagnosticWriteDataBy IdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticWriteDataBy Identifier in the given context.

Table A.165: DiagnosticWriteDataByIdentifier

Enumeration	**DoIpEidRetrievalEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	Enumeration with options to retrieve EID.
Aggregated by	DoIpNetworkConfiguration.eidRetrieval, DoIpNetworkConfigurationDesign.eidRetrieval
Literal	*Description*
eidUseApi	API DiagnosticDoIPEntityIdentification is used to retrieve eid **Tags:** atp.EnumerationLiteralIndex=1
eidUseConfigValue	eid is configured manually by DoIpInstantiation.eid **Tags:** atp.EnumerationLiteralIndex=2
eidUseMac	MAC of the network interface is used as eid **Tags:** atp.EnumerationLiteralIndex=0

Table A.166: DoIpEidRetrievalEnum

Class	**DoIpFunctionalClusterDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class defines the attributes for the DoIP configuration settings in the MachineDesign.			
Base	*ARObject*, *AbstractFunctionalClusterDesign*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
doIpLogical Address	DoIpLogicalAddress	0..1	aggr	This aggregation contains information about the DoIP logical address.
doIpProtocol Version	PositiveInteger	0..1	attr	Configures the DoIP protocol version used in the generic DoIP header. The valid range of this parameter is defined by the always latest release of ISO 13400-2 and can be extended with every new release of the ISO document. As example a value of 0x03 defines the ISO 13400-2:2019 release.
eid	PositiveUnlimitedInteger	0..1	attr	Configured EID (Entity ID) used for VehicleIdentification Request.
entityStatusMax ByteFieldUse	Boolean	0..1	attr	This attribute is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.
maxRequest Bytes	PositiveInteger	0..1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
network Interface	DoIpNetwork ConfigurationDesign	*	aggr	This element collects DoIP properties that are network interface specific.

▽

△

Class	DoIpFunctionalClusterDesign			
request Configuration Design	DoIpRequest ConfigurationDesign	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected.

Table A.167: DoIpFunctionalClusterDesign

Class	DoIpInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
doIpDesign	DoIpFunctionalCluster Design	0..1	ref	Reference to the DoIP Design that this DoIpInstantiation implements.
gid	PositiveUnlimitedInteger	0..1	attr	Configured GID (Group ID) used for VehicleIdentification Request. If configured, take this value (and set "Further action required" byte to 0x00="No further action required"), if not configured use ServiceInterface Do IPGroupIdentification to retrieve GID and 'further action required' values.
logicalAddress	PositiveInteger	0..1	attr	Describes the logical address of the DoIP entity, which is used for VehicleAnnouncement and RoutingActivation responses.
network Interface	DoIpNetwork Configuration	*	aggr	Network interface specific DoIP properties.
request Configuration	DoIpRequest Configuration	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected.

Table A.168: DoIpInstantiation

Class	DoIpNetworkConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This element collects DoIP properties that are network interface specific.			
Base	*ARObject*			
Aggregated by	DoIpInstantiation.networkInterface			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eidRetrieval	DoIpEidRetrievalEnum	0..1	attr	This attribute defines how DoIp Entitiy Identification is retrieved.
isActivationLine Dependent	Boolean	0..1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.

▽

△

Class	**DoIpNetworkConfiguration**			
maxInitial Vehicle Announcement Time	TimeValue	0..1	attr	Upper bound for the time to wait in [s] for sending first vehicle announcement message after IP address assignment. Represents parameter A_DoIP_Announce_ Wait of ISO 13400-2:2019. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN.
network Configuration Design	DoIpNetwork ConfigurationDesign	0..1	ref	Reference to the DoIP network configuration design that this DoIpNetworkConfiguration implements.
network InterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2019.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_ TCP_General_Inactivity of ISO 13400-2:2019.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2019.
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_ Announce_Num of ISO 13400-2:2019.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2019.
vehicle Identification SyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table A.169: DoIpNetworkConfiguration

Class	**DoIpNetworkConfigurationDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This element collects DoIP properties that are network interface specific.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DoIpFunctionalClusterDesign.networkInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eidRetrieval	DoIpEidRetrievalEnum	0..1	attr	This attribute defines how DoIp Entitiy Identification is retrieved.
isActivationLine Dependent	Boolean	0..1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.

▽

△

Class	**DoIpNetworkConfigurationDesign**			
maxInitial Vehicle Announcement Time	TimeValue	0..1	attr	Upper bound for the time to wait in [s] for sending first vehicle announcement message after IP address assignment. Represents parameter A_DoIP_Announce_ Wait of ISO 13400-2:2019. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].
maxTester Connections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN.
network InterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2019.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_ TCP_General_Inactivity of ISO 13400-2:2019.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2019.
tpConnection	GenericTpConnection	*	ref	Reference to a TpConnection that identifies the receiver(s) of a particular communication
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_ Announce_Num of ISO 13400-2:2019.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2019.
vehicle Identification SyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table A.170: DoIpNetworkConfigurationDesign

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	***Identifiable*** (abstract)			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractFunctionalClusterDesign*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstance Filter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternal Behavior, ApApplicationEndpoint, *ApmcAbstractDefinition*, *ApmcConfigurationElementDef*, *Apmc ContainerElementValue*, ApmcContainerValue, ApmcEnumerationLiteralDef, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildAction Entity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientId Definition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *Comm ConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoKeySlotDesign, CryptoKeySlotUsageDesign, CryptoProvider, *CryptoServiceMapping*, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, Data Transformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic FunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovd MethodPrimitive, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIp LogicalAddress, DoIpNetworkConfigurationDesign, DoIpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, Ethernet WakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMap Element, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForget MethodMapping, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, Global Supervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBus Part*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, Method Mapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterfaceElement*, *Phm Supervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwComposition Prototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecution Context, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInterfaceElementSecureComConfig, *Service MethodDeployment*, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalServiceTranslation Props, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, *StateManagementActionItem*, State ManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, Static SocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionMode Condition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwitchAsynchronous TrafficShaperGroupEntry, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSync Accuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, Timing ModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40

▽

△

Class	***Identifiable*** (abstract)			
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table A.171: Identifiable

Class	***MultilanguageReferrable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders). They also may have a longName. But they are not considered to contribute substantially to the overall structure of an AUTOSAR description. In particular it does not contain other Referrables.			
Base	*ARObject*, *Referrable*			
Subclasses	Caption, DefItem, DocumentationContext, *Identifiable*, SdgCaption, *TraceReferrable*, *Traceable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
longName	MultilanguageLong Name	0..1	aggr	This specifies the long name of the object. Long name is targeted to human readers and acts like a headline.

Table A.172: MultilanguageReferrable

Class	PlatformModuleEthernetEndpointConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the configuration of a port, protocol type and IP address (local address) of the communication on a VLAN. **Tags:** atp.recommendedPackage=PlatformModuleEndpointConfigurations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PlatformModuleEndpointConfiguration*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector (VLAN) for which the network configuration is defined.
remoteConfig	RemoteEndpoint Configuration	*	aggr	Defintion of remote addresses of peers.
secureCom PropsForTcp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration.
secureCom PropsForUdp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration.
tcpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a tcp port number.
udpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a udp port number.

Table A.173: PlatformModuleEthernetEndpointConfiguration

Class	***PortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallStateSwitchInterface, *IdsmAbstractPort Interface*, LogAndTraceInterface, ModeSwitchInterface, NetworkManagementPortInterface, *Persistency Interface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=namespace.shortName

Table A.174: PortInterface

Class	***PortPrototype*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.

▽

△

Class	**PortPrototype** (abstract)			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.175: PortPrototype

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table A.176: RPortPrototype

Class	**Referrable** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescription Entity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, Pnc MappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, Someip RequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	*Referrable* (abstract)			
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table A.177: Referrable

Class	**SoftwareCluster**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.recommendedPackage=SoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
artifact Checksum	ArtifactChecksum	*	aggr	This aggregation carries the checksums for artifacts contained in the enclosing SoftwareCluster. Please note that the value of these checksums is only applicable at the time of configuration. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=artifactChecksum.shortName, artifact Checksum.uri
artifactLocator	ArtifactLocator	*	aggr	This aggregation represents the artifact locations that are relevant in the context of the enclosing SoftwareCluster
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedARElement
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster.
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster.

▽

△

Class	**SoftwareCluster**			
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef
diagnostic Deployment Props	SoftwareCluster DiagnosticDeployment Props	0..1	ref	This reference identifies the applicable SoftwareCluster DiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster.
installation Behavior	SoftwareCluster InstallationBehavior Enum	0..1	attr	This attribute controls the behavior of the SoftwareCluster in terms of installation.
license	Documentation	*	ref	This attribute allows for the inclusion of the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license.
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test.
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature.
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table A.178: SoftwareCluster

Class	***SoftwareClusterDiagnosticAddress*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address in an abstract form. Sub-classes are supposed to clarify how the diagnostic address shall be defined according to the applicable addressing scheme (DoIP vs. CAN TP vs. ...).			
Base	*ARObject*			
Subclasses	SoftwareClusterSovdAddress, SoftwareClusterUdsDiagnosticAddress			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
address Semantics	SoftwareCluster DiagnosticAddress SemanticsEnum	0..1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table A.179: SoftwareClusterDiagnosticAddress

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster.
Aggregated by	*SoftwareClusterDiagnosticAddress*.addressSemantics
Literal	***Description***
functionalAddress	This address represents a functional address. **Tags:** atp.EnumerationLiteralIndex=1
physicalAddress	This address represents a physical address. **Tags:** atp.EnumerationLiteralIndex=0

Table A.180: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	**SoftwareClusterDiagnosticDeploymentProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class acts as the owner of all deployment-related diagnostic properties of a SoftwareCluster. **Tags:** atp.recommendedPackage=SoftwareClusterDiagnosticProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference identifies the DiagnosticContributionSet that is applicable for the referencing SoftwareCluster.
max Conversations	PositiveInteger	0..1	attr	Maximum number of diagnostic Conversations supported by the Diagnostic Server Instance. This attribute has no relation to the definition of the maximum number of clients in DoIP context, configured by means of DoIpNetwork Configuration.maxTesterConnections.
powerDown Time	PositiveInteger	0..1	attr	This attribute indicates the minimum time of the stand-by sequence the server will remain in the power-down sequence. The unit is seconds.
validation Configuration	DiagnosticService ValidationConfiguration	0..1	aggr	This aggregation represents the ability to define the order of manufacturer and supplier validations in diagnostic management.

Table A.181: SoftwareClusterDiagnosticDeploymentProps

Class	**SoftwareClusterSovdAddress**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address specifically for the SOVD case. **Tags:** atp.Status=candidate			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component Qualifier	String	0..1	attr	This attribute is used to specify the component qualifier for the usage in an SOVD query. **Tags:** atp.Status=candidate

Table A.182: SoftwareClusterSovdAddress

Class	***SovdModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This abstract meta-class serves as the base class for meta-classes that describe the configuration of an SOVD module. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	SovdGatewayInstantiation, SovdServerInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communication Connector	Communication Connector	*	ref	This reference identifies the collection of communication connectors used by the SOVD module instantiation for vehicle-internal communication. **Tags:** atp.Status=candidate
nodeIdentifier	String	0..1	attr	This attribute represents the local hostname of the SOVD server to be used during the execution of the DNS-SD protocol. **Tags:** atp.Status=candidate
securePropsFor Tcp	SecureComProps	0..1	ref	This reference is used to identify the applicable TCP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Status=candidate
securePropsFor Udp	SecureComProps	0..1	ref	This reference is used to identify the applicable UDP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Status=candidate

Table A.183: SovdModuleInstantiation

Class	**SovdServerInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This meta-class represents the configuration of an SOVD server. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *SovdModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
component Qualifier	String	0..1	attr	This attributes described the component qualifier used to compose an SOVD query. **Tags:** atp.Status=candidate

Table A.184: SovdServerInstantiation

Class	«atpVariation» **SwDataDefProps**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddr Method, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalid Value • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime
Base	*ARObject*
Aggregated by	*AutosarDataType*.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, Cpp ImplementationDataTypeElement.swDataDefProps, *DataPrototype*.swDataDefProps, DataPrototype TransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, Diagnostic EnvDataElementCondition.swDataDefProps, DltArgument.networkRepresentation, FlatInstance Descriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDef Props.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, *ReceiverComSpec*.network Representation, SecurityEventContextDataElement.networkRepresentation, *SenderComSpec*.network Representation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTarget Props.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, System Signal.physicalProps

Attribute	***Type***	***Mult.***	***Kind***	***Note***
additionalNative TypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** xml.sequenceOffset=180

▽

△

Class	«atpVariation» **SwDataDefProps**			
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. **Tags:** xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90

▽

△

Class	«atpVariation» **SwDataDefProps**			
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290

▽

△

Class	«atpVariation» **SwDataDefProps**			
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** xml.sequenceOffset=355

Table A.185: SwDataDefProps

Class	**SymbolProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	*ARObject*, *ImplementationProps*, *Referrable*			
Aggregated by	Allocator.namespace, ApApplicationErrorDomain.namespace, *AtomicSwComponentType*.symbolProps, *CppImplementationDataType*.namespace, ImplementationDataType.symbolProps, *PortInterface*.namespace, SecurityEventDefinition.eventSymbolName			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.186: SymbolProps

Class	**ValueGroup**			
Package	M2::MSR::CalibrationData::CalibrationValue			
Note	This element enables values to be grouped. It can be used to perform row and column-orientated groupings, so that these can be rendered properly e.g. as a table.			
Base	*ARObject*			
Aggregated by	SwValues.vg			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
label	MultilanguageLong Name	0..1	aggr	This label allows to give the valueGroup a particular name. It can be used if the Values are rendered as a table. **Tags:** xml.sequenceOffset=20
vgContents	SwValues	0..1	aggr	This represents the contents of the value group. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table A.187: ValueGroup

B Demands and constraints on Base Software (normative)

This functional cluster defines no demands or constraints for the Base Software on which the AUTOSAR Adaptive Platform is running on (usually a POSIX-compatible operating system).

C Platform Extension Interfaces (normative)

This chapter provides a reference of the Platform Extension Interfaces defined by this functional cluster. Platform Extension Interfaces are intended to be used/provided by an OEM or Integrator to extend the functionality of the AUTOSAR Adaptive Platform.

C.1 Header: apext/diag/uds_transport/protocol_handler.h

C.1.1 Class: UdsTransportProtocolHandler

[SWS_DM_00315] Definition of API class apext::diag::uds_transport::UdsTransportProtocolHandler

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	class
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Forwarding header file:	#include "apext/diag/diag_fwd.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsTransportProtocolHandler
Syntax:	`class UdsTransportProtocolHandler {...};`
Description:	Abstract Class, which a specific UDS Transport Protocol (plugin) shall subclass.

⌋

C.1.1.1 Public Member Types

C.1.1.1.1 Enumeration: InitializationResult

[SWS_DM_09017] Definition of API enum apext::diag::uds_transport::UdsTransportProtocolHandler::InitializationResult

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	enumeration
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Forwarding header file:	#include "apext/diag/diag_fwd.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`
Symbol:	InitializationResult

▽

△

Underlying type:	std::uint8_t	
Syntax:	`enum class InitializationResult : std::uint8_t {...};`	
Values:	kInitializeOk= 0	--
	kInitializeFailed= 1	--
Description:	Result of Initialize handler.	

⌋

C.1.1.2 Protected Member Variables

C.1.1.2.1 transportprotocolManager

[SWS_DM_09025] Definition of API variable apext::diag::uds_transport::UdsTransportProtocolHandler::transportprotocolManager

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	variable
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`
Symbol:	transportprotocolManager
Type:	`UdsTransportProtocolMgr &`
Syntax:	`UdsTransportProtocolMgr& transportprotocolManager;`
Description:	The UdsTransportProtocolMgr used/provided by the DM/DCM.
Visibility:	protected

⌋

C.1.1.3 Public Member Functions

C.1.1.3.1 Special Member Functions

C.1.1.3.1.1 Destructor

[SWS_DM_09016] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::~UdsTransportProtocolHandler

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`
Syntax:	`virtual ~UdsTransportProtocolHandler ()=default;`
Thread Safety:	implementation defined
Description:	Destructor of UdsTransportProtocolHandler.

⌋

C.1.1.3.2 Constructors

C.1.1.3.2.1 UdsTransportProtocolHandler

[SWS_DM_09015] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`	
Syntax:	`explicit UdsTransportProtocolHandler (const UdsTransportProtocolHandlerID handlerId, UdsTransportProtocolMgr &transportProtocolMgr) noexcept;`	
Parameters (in):	handlerId	the handler ID used by DM to identify this handler. This is just a number/identification given by the DM core when instantiating a UdsTransportProtocolHandler instance to be able to distinguish it from other handler-plugins or built-in UdsTransportProtocolHandler implementations.
	transportProtocolMgr	reference to UdsTransportProtocolMgr owned by this DM, with which UdsTransportProtocolHandler instance shall interact.
Exception Safety:	exception safe	

▽

△

Thread Safety:	implementation defined
Description:	Constructor of UdsTransportProtocolHandler.

⌋

C.1.1.3.3 Member Functions

C.1.1.3.3.1 GetHandlerID

[SWS_DM_00325] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::GetHandlerID

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`	
Syntax:	`virtual UdsTransportProtocolHandlerID GetHandlerID () const noexcept;`	
Return value:	UdsTransportProtocol HandlerID	UdsTransportProtocolHandlerID.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Return the UdsTransportProtocolHandlerID, which was given to the implementation during construction (ctor call).	

⌋

C.1.1.3.3.2 GetPeriodicHandler

[SWS_DM_01068] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::GetPeriodicHandler

Upstream requirements: RS_Diag_04215

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`

▽

△

Syntax:	`virtual ara::core::Result< UdsTransportProtocolPeriodicHandler & > GetPeriodicHandler () noexcept;`	
Return value:	ara::core::Result< UdsTransportProtocolPeriodicHandler & >	UdsTransportProtocolPeriodicHandler reference if periodic transmissions are supported on this transport protocol, an error if not supported
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Returns the corresponding periodic TP handler.	

⌋

C.1.1.3.3.3 Initialize

[SWS_DM_00319] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::Initialize

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`	
Syntax:	`virtual InitializationResult Initialize () noexcept=0;`	
Return value:	InitializationResult	kInitializeOk if initialization was successful, else kInitializeFailed.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Initializes UDS transport protocol handler. This methods needs to be called before a call of Start(). It is meant to separate the actual initialization of the transport protocol handler from the constructor.	

⌋

C.1.1.3.3.4 NotifyReestablishment

[SWS_DM_00326] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`	
Syntax:	`virtual bool NotifyReestablishment (ChannelID channelId) noexcept=0;`	
Parameters (in):	channelId	channelID, whose re-establishment shall be notified to Uds TransportProtocolMgr
Return value:	bool	true if notification request is accepted and can be fulfilled.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Tells the UdsTransportProtocolHandler, to notify the DM core via UdsTransportProtocol Mgr::ChannelReestablished if the given channel has been re-established after next Uds TransportProtocolHandler::Start. The main purpose of this method is to allow the DM to provide an ECU-Reset (0x11 service), with configuration option "positive response AFTER reset". In this scenario the request for 0x11 will be received on a certain channel (GlobalChannelIdentifier). Then the ECU-Reset takes place and after ECU-Restart all UdsProtocolHandlers/plugins get restarted via call to UdsTransportProtocolHandler::Start(). Now there are two expectations, when this method has been called before and returned "true": IF the same remote client connects to the UdsProtocolHandler, it shall get a channel identification with the same identifying as the last time. It consecutively calls UdsTransportProtocolMgr::Channel Reestablished(GlobalChannelIdentifier).	
Notes:	: The UdsTransportProtocolHandler will call UdsTransportProtocolMgr::ChannelReestablished() after UdsTransportProtocolHandler::Start() is called and as soon as it detects that the remote client/tester is reachable again. The detection/decision, whether the "same" client reconnects as before is an UdsProtocol Handler implementation specific decision. The general expectation is: If the channel is set up from exactly the same remote network-endpoint, it typically gets the same channelID. To support this functionality the implementation at least has to store non-volatile, that this notification has to be done. Further it might be needed to store some additional connection specific info non-volatile to make sure, that the same channelID can be reassigned. This is the case if the mapping of protocol specific channel info -> channelID isn't a stable bijective mapping! Small example: The underlying network protocol, which UdsProtocolHandler implements is based on TCP. At the point in time, where the 0x11 SI request is received on channel identified by the DM calls NotifyReestablishment() on this channelID. Now the implementation of UdsProtocolHandler stores non- volatile in the context of this call: the Network Endpoint (IP-address and port number) of the channel the NetworkEndpoint (IP-address and port number) of the local port (because in this example, the UdsTransportProtocolHandler listens on/supports different ports) the channelID it has currently assigned. After restart this channelID is only reused for a channel with exactly the same NetworkEndpoint addresses as stored non-volatile. If this channelID then gets reassigned, then UdsTransportProtocol Mgr::ChannelReestablished() has to be called.	

⌋

C.1.1.3.3.5 Start

[SWS_DM_00322] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::Start

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`
Syntax:	`virtual void Start () noexcept=0;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Start processing the Uds Transport Protocol. Within the implementation there might be some stack specific implementation. The method is asynchronous to allow the DM to start multiple UdsTransportProtocolHandler in parallel to reduce startup time.

⌋

C.1.1.3.3.6 Stop

[SWS_DM_00323] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::Stop

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`
Syntax:	`virtual void Stop () noexcept=0;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Method to indicate that this UdsTransportProtocolHandler should terminate. If UdsTransportProtocolHandler has stopped, it shall call UdsTransportProtocolMgr::Handler Stopped(UdsTransportProtocolHandlerID) After return from Stop(), the handler-plugin shall NOT call to UdsTransportProtocolMgr with any other method but UdsTransportProtocolMgr::HandlerStopped()

⌋

C.1.1.3.3.7 Transmit

[SWS_DM_00327] Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::Transmit

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolHandler`	
Syntax:	`virtual void Transmit (UdsMessageConstPtr message, ChannelID channelId) noexcept=0;`	
Parameters (in):	message	The message to be transmitted as a UdsMessage::Ptr (unique_ptr style). UdsTransportProtocolHandler has to give back this UdsMessage::Ptr via UdsTransportProtocolMgr::TransmitConfirmation() to signal, that it is done with this message.
	channelId	identification of channel on which to transmit.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Transmit a Uds message via the underlying Uds Transport Protocol channel. This transmit API covers T_Data.req of ISO 14229-2 Figure 2.	

⌋

C.2 Header: apext/diag/uds_transport/protocol_mgr.h

C.2.1 Class: UdsTransportProtocolMgr

[SWS_DM_00306] Definition of API class apext::diag::uds_transport::UdsTransportProtocolMgr

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	class
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"
Forwarding header file:	#include "apext/diag/diag_fwd.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsTransportProtocolMgr
Syntax:	`class UdsTransportProtocolMgr {...};`
Description:	--

⌋

C.2.1.1 Public Member Types

C.2.1.1.1 Type Alias: GlobalChannelIdentifier

[SWS_DM_09021] Definition of API type apext::diag::uds_transport::UdsTransportProtocolMgr::GlobalChannelIdentifier

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`
Symbol:	GlobalChannelIdentifier
Syntax:	`using GlobalChannelIdentifier = std::tuple<UdsTransportProtocolHandlerID, ChannelID>;`
Description:	Type of tuple to pack UdsTransportProtocolHandlerID and ChannelID together, to form a global unique (among all used UdsTransportProtocolHandlers within DM) identifier of a UdsTransport Protocol channel.

⌋

C.2.1.1.2 Enumeration: IndicationResult

[SWS_DM_00384] Definition of API enum apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	enumeration	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Forwarding header file:	#include "apext/diag/diag_fwd.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Symbol:	IndicationResult	
Underlying type:	std::uint8_t	
Syntax:	`enum class IndicationResult : std::uint8_t {...};`	
Values:	kIndicationOk= 0	--
	kIndicationOccupied= 1	--
	kIndicationOverflow= 2	--
	kIndicationUnknown TargetAddress= 3	--
Description:	Allowed return values from the DM Server to the Transport Protocol Handler Instance for the received IndicateMessage.	

⌋

C.2.1.1.3 Enumeration: TransmissionResult

[SWS_DM_00307] Definition of API enum apext::diag::uds_transport::UdsTransportProtocolMgr::TransmissionResult

Upstream requirements: RS_Diag_04172, RS_Diag_04147, RS_Diag_04168

⌈

Kind:	enumeration	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Forwarding header file:	#include "apext/diag/diag_fwd.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Symbol:	TransmissionResult	
Underlying type:	std::uint8_t	
Syntax:	`enum class TransmissionResult : std::uint8_t {...};`	
Values:	kTransmitOk= 0	--
	kTransmitFailed= 1	--
Description:	Values denoting the Transmission Confirmation Result provided by the Transport protocol Handler Instance to the DM Server.	

⌋

C.2.1.2 Public Member Functions

C.2.1.2.1 Special Member Functions

C.2.1.2.1.1 Destructor

[SWS_DM_01524] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::~UdsTransportProtocolMgr

Upstream requirements: RS_AP_00134

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`
Syntax:	`virtual ~UdsTransportProtocolMgr () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor of class UdsTransportProtocolMgr.

⌋

C.2.1.2.2 Member Functions

C.2.1.2.2.1 ChannelReestablished

[SWS_DM_00313] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void ChannelReestablished (GlobalChannelIdentifier globalChannelId) noexcept=0;`	
Parameters (in):	globalChannelId	transport protocol channel, which is available again.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	notification call from the given transport channel, that it has been reestablished since the last (Re)Start from the UdsTransportProtocolHandler to which this channel belongs. To activate this notification a previous call to UdsTransportProtocolHandler::NotifyReestablishment() has to be done. See further documentation at UdsTransportProtocolHandler::NotifyReestablishment().	

⌋

C.2.1.2.2.2 HandleMessage

[SWS_DM_00311] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void HandleMessage (UdsMessagePtr message) noexcept=0;`	
Parameters (in):	message	The Uds message ptr (unique_ptr semantics) with the request. Ownership of the UdsMessage is given back to the generic DM core here.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	

▽

△

Description:	Hands over a valid received Uds message (currently this is only a request type) from transport layer to session layer. It corresponds to T_Data.ind of Figure 2 from ISO 14229-2. The behavior is asynchronously. I.e. the UdsMessage is handed over to Session Layer and it is expected, that it "instantly" returns, which means, that real processing of the message shall be done asynchronously!

⌋

C.2.1.2.2.3 HandlerStopped

[SWS_DM_00314] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void HandlerStopped (UdsTransportProtocolHandlerID handlerId) noexcept=0;`	
Parameters (in):	handlerId	indication, which plugin stopped.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	notification from handler, that it has stopped now (e.g. closed down network connections, freed resources, etc...) This callback is expected as a reaction from handler to a call to UdsTransportProtocol Handler::Stop.	

⌋

C.2.1.2.2.4 IndicateMessage

[SWS_DM_00309] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual std::pair< IndicationResult, UdsMessagePtr > IndicateMessage ( UdsMessage::Address sourceAddr, UdsMessage::Address targetAddr, UdsMessage::TargetAddressType type, GlobalChannelIdentifier globalChannelId, std::size_t size, ara::core::Span< const std::uint8_t > payloadInfo) noexcept=0;`	
Parameters (in):	sourceAddr	UDS source address of message
	targetAddr	UDS target address of message
	type	indication whether its is phys/func request
	globalChannelId	transport protocol channel on which message start happened
	size	size in bytes of the UdsMessage starting from SID.
	payloadInfo	View onto the first received payload bytes, if any. This view shall be used only within this function call. It is recommended that the TP provides at least the first two bytes of the request message, so the DM can identify a functional TesterPresent, Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
Return value:	std::pair< IndicationResult, UdsMessagePtr >	Pair of IndicationResult and a pointer to UdsMessage owned/ created by DM core and returned to the handler to get filled.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Indicates a message start. This is an interface, which is just served/called by UdsTransport ProtocolHandlers, which return true from UdsTransportProtocolHandlers::isStartOfMessage IndicationSupported().	

⌋

C.2.1.2.2.5 NotifyMessageFailure

[SWS_DM_00310] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void NotifyMessageFailure (UdsMessagePtr message) noexcept=0;`	
Parameters (in):	message	the pointer to UdsMessage handed back over to the session layer.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Indicates, that the message indicated via IndicateMessage() has failure and will not lead to a final HandleMessage() call.	

⌋

C.2.1.2.2.6 PeriodicTransmitConfirmation

[SWS_DM_01069] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation

Upstream requirements: RS_Diag_04215

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void PeriodicTransmitConfirmation (ara::core::Vector< UdsMessageConstPtr > messages, std::size_t numberOfSentMessages) noexcept=0;`	
Parameters (in):	messages	The same ordered list of messages previously passed to UdsTransportProtocolPeriodicHandler::PeriodicTransmit.
	numberOfSentMessages	The number of successfully sent messages from the "messages" list.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Confirmation of sent messages and number.	

⌋

C.2.1.2.2.7 TransmitConfirmation

[SWS_DM_00312] Definition of API function apext::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation

Upstream requirements: RS_Diag_04172, RS_Diag_04147, RS_Diag_04168

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolMgr`	
Syntax:	`virtual void TransmitConfirmation (UdsMessageConstPtr message, TransmissionResult result) noexcept=0;`	
Parameters (in):	message	for which message (created in IndicateMessage()) this is the confirmation.
	result	Result of transmission. In case UDS message could be transmitted on network layer: kTransmitOk), kTransmitFailed else.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	notification about the outcome of a transmit request called by core DM at the handler via Uds TransportProtocolHandler::Transmit This transmit API covers T_Data.con of ISO 14229-2 Figure 2.	

⌋

C.3 Header: apext/diag/uds_transport/protocol_periodic_handler.h

C.3.1 Class: UdsTransportProtocolPeriodicHandler

[SWS_DM_01064] Definition of API class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler

Upstream requirements: RS_Diag_04215

⌈

Kind:	class
Header file:	#include "apext/diag/uds_transport/protocol_periodic_handler.h"
Forwarding header file:	#include "apext/diag/diag_fwd.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsTransportProtocolPeriodicHandler
Syntax:	`class UdsTransportProtocolPeriodicHandler {...};`
Description:	UdsTransportProtocolPeriodicHandler class to support 0x2A service from ISO.

⌋

C.3.1.1 Public Member Functions

C.3.1.1.1 Special Member Functions

C.3.1.1.1.1 Destructor

[SWS_DM_02089] Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::~UdsTransportProtocolPeriodicHandler

Upstream requirements: RS_Diag_04215

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_periodic_handler.h"
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler`
Syntax:	`virtual ~UdsTransportProtocolPeriodicHandler () noexcept;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor.

⌋

C.3.1.1.2 Member Functions

C.3.1.1.2.1 GetMaxPayloadLength

[SWS_DM_01066] Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength

Upstream requirements: RS_Diag_04215

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_periodic_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler`	
Syntax:	`virtual std::size_t GetMaxPayloadLength (ChannelID channelId) const noexcept=0;`	
Parameters (in):	channelId	The concrete connection for which the maximum payload length is reported.
Return value:	std::size_t	supported payload length
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Reports the maximum payload length supported for a single periodic transmission on the channel.	

⌋

C.3.1.1.2.2 GetNumberOfPeriodicMessages

[SWS_DM_01065] Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages

Upstream requirements: RS_Diag_04215

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_periodic_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler`	
Syntax:	`virtual std::size_t GetNumberOfPeriodicMessages (ChannelID channelId) const noexcept=0;`	
Parameters (in):	channelId	The concrete connection to send over
Return value:	std::size_t	number of periodic messages
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Reports the TP implementation and connection specific number of periodic messages.	

⌋

C.3.1.1.2.3 PeriodicTransmit

[SWS_DM_01067] Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit

Upstream requirements: RS_Diag_04215

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_periodic_handler.h"	
Scope:	`class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler`	
Syntax:	`virtual void PeriodicTransmit (ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages) noexcept=0;`	
Parameters (in):	channelId	The concrete connection to send over
	messages	Ordered list of messages to send at once
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Sends all the messages in the list in the given order. If one message transmission fails, the send process is stopped and the PeriodicTransmitConfirmation is invoked with the number of sent messages and the same Vector with UdsMessages.	

⌋

C.4 Header: apext/diag/uds_transport/protocol_types.h

C.4.1 Non-Member Types

C.4.1.1 Type Alias: ByteSpan

[SWS_DM_00338] Definition of API type apext::diag::uds_transport::ByteSpan

Upstream requirements: RS_Diag_04147

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/protocol_types.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	ByteSpan
Syntax:	`using ByteSpan = ara::core::Span<ara::core::Byte>;`
Description:	This is the type of ByteSpan.

⌋

C.4.1.2 Type Alias: ChannelID

[SWS_DM_00337] Definition of API type apext::diag::uds_transport::ChannelID

Upstream requirements: RS_Diag_04147

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/protocol_types.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	ChannelID
Syntax:	`using ChannelID = std::uint32_t;`
Description:	This is the identification of channel on which to transmit.

⌋

C.4.1.3 Type Alias: UdsTransportProtocolHandlerID

[SWS_DM_00336] Definition of API type apext::diag::uds_transport::UdsTransportProtocolHandlerID

Upstream requirements: RS_Diag_04147, RS_Diag_04168

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/protocol_types.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsTransportProtocolHandlerID
Syntax:	`using UdsTransportProtocolHandlerID = std::uint8_t;`
Description:	UdsTransportProtocolHandler are flexible "plugins", which need an identification.

⌋

C.5 Header: apext/diag/uds_transport/uds_message.h

C.5.1 Non-Member Types

C.5.1.1 Type Alias: UdsMessageConstPtr

[SWS_DM_00304] Definition of API type apext::diag::uds_transport::UdsMessageConstPtr

Upstream requirements: RS_Diag_04147

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsMessageConstPtr
Syntax:	`using UdsMessageConstPtr = std::unique_ptr<const UdsMessage, std::function<void(const UdsMessage*)>>;`
Description:	This is the unique_ptr for constant UdsMessages containing a custom deleter as provided by the generic/core DM part towards the UdsTransportLayer-Plugin.
Notes:	How the exact typedef for UdsMessageConstPtr looks like, is up to the DM product vendor. I.e. how f.i. the deleter signature looks like ... basically the minimal agreement is: UdsMessage ConstPtr shall behave like a std::unique_ptr<const UdsMessage>!

⌋

C.5.1.2 Type Alias: UdsMessagePtr

[SWS_DM_00303] Definition of API type apext::diag::uds_transport::UdsMessagePtr

Upstream requirements: RS_Diag_04147

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsMessagePtr
Syntax:	`using UdsMessagePtr = std::unique_ptr<UdsMessage, std::function<void(UdsMessage*)>>;`
Description:	This is the unique_ptr for UdsMessages containing a custom deleter as provided by the generic/ core DM part towards the UdsTransportLayer-Plugin.
Notes:	How the exact typedef for UdsMessagePtr looks like, is up to the DM product vendor. I.e. how f.i. the deleter signature looks like ... basically the minimal agreement is: UdsMessagePtr shall behave like a std::unique_ptr<UdsMessage>!

⌋

C.5.2 Class: UdsMessage

[SWS_DM_00291] Definition of API class apext::diag::uds_transport::UdsMessage

Upstream requirements: RS_Diag_04147

⌈

Kind:	class
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Forwarding header file:	#include "apext/diag/diag_fwd.h"
Scope:	`namespace apext::diag::uds_transport`
Symbol:	UdsMessage
Syntax:	`class UdsMessage {...};`
Description:	class represents an UDS message exchanged between DM generic core (UdsTransportProtocol Mgr) and a specific implementation of UdsTransportProtocolHandler on diagnostic request reception path or diagnostic response transmission path. UdsMessage provides the storage for UDS requests/responses. Instances of UdsMessage (with optimized resource allocation) are only created by DM generic core. UdsTransportProtocol Handler read/write on it.

⌋

C.5.2.1 Public Member Types

C.5.2.1.1 Type Alias: Address

[SWS_DM_00293] Definition of API type apext::diag::uds_transport::UdsMessage::Address

Upstream requirements: RS_Diag_04147

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`class apext::diag::uds_transport::UdsMessage`
Symbol:	Address
Syntax:	`using Address = std::uint16_t;`
Description:	type for UDS source and target addresses

⌋

C.5.2.1.2 Type Alias: MetaInfoMap

[SWS_DM_00294] Definition of API type apext::diag::uds_transport::UdsMessage::MetaInfoMap

Upstream requirements: RS_Diag_04147, RS_Diag_04170

⌈

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`class apext::diag::uds_transport::UdsMessage`
Symbol:	MetaInfoMap
Syntax:	`using MetaInfoMap = ara::core::Map<ara::core::String, ara::core::String>;`
Description:	Type for the meta information attached to a UdsMessage. .

⌋

C.5.2.1.3 Enumeration: TargetAddressType

[SWS_DM_00296] Definition of API enum apext::diag::uds_transport::UdsMessage::TargetAddressType

Upstream requirements: RS_Diag_04147

⌈

Kind:	enumeration	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Forwarding header file:	#include "apext/diag/diag_fwd.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Symbol:	TargetAddressType	
Underlying type:	std::uint8_t	
Syntax:	`enum class TargetAddressType : std::uint8_t {...};`	
Values:	kPhysical= 0	--
	kFunctional= 1	--
Description:	type of target address in UdsMessage	

⌋

C.5.2.2 Public Member Functions

C.5.2.2.1 Special Member Functions

C.5.2.2.1.1 Destructor

[SWS_DM_09010] Definition of API function apext::diag::uds_transport::UdsMessage::~UdsMessage

Upstream requirements: RS_Diag_04147

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`class apext::diag::uds_transport::UdsMessage`
Syntax:	`virtual ~UdsMessage ()=default;`
Thread Safety:	implementation defined
Description:	Destructing the uds message.

⌋

C.5.2.2.2 Member Functions

C.5.2.2.2.1 AddMetaInfo

[SWS_DM_00302] Definition of API function apext::diag::uds_transport::UdsMessage::AddMetaInfo

Upstream requirements: RS_Diag_04147, RS_Diag_04170

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual void AddMetaInfo (MetaInfoMap metaInfo) noexcept;`	
Parameters (in):	metaInfo	meta information relevant for UdsMessage
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	unsafe	
Description:	add new metaInfo to this message.	
Notes:	typically called by the transport plugin to add channel specific meta-info. (see SWS - there are already predefined meta-info keys for DoIP....)	

⌋

C.5.2.2.2.2 GetPayload

[SWS_DM_00301] Definition of API function apext::diag::uds_transport::UdsMessage::GetPayload

Status: DRAFT

Upstream requirements: RS_Diag_04147

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual uds_transport::ByteSpan & GetPayload () noexcept;`	
Return value:	uds_transport::ByteSpan &	payload of the UDSMessage starting from SID.
Exception Safety:	exception safe	
Thread Safety:	unsafe	
Description:	return the underlying buffer for write access.	

▽

△

Notes:	needed by UdsTransportProtocolHandler impl. to fill the UdsMessage with data in RX path. I.e. UdsTransportProtocolHandler impl. gets the UdsMessage instance from call to UdsTransport ProtocolMgr::IndicateMessage() and then calls this method on it and write into returned uds_ transport::ByteVector. marked as "unsafe" with regard to threadsafety as implementation is allowed to do ressource allocation of buffer in the context of this call.

⌋

C.5.2.2.2.3 GetPayload

[SWS_DM_00300] Definition of API function apext::diag::uds_transport::Uds Message::GetPayload

Status: DRAFT
Upstream requirements: RS_Diag_04147

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual const ara::core::Vector< const std::uint8_t > & GetPayload () const noexcept;`	
Return value:	const ara::core::Vector< const std::uint8_t > &	The entire payload (A_Data) with no write access.
Exception Safety:	exception safe	
Thread Safety:	unsafe	
Description:	Get the UDS message data starting with the SID (A_Data as per ISO) with no write access.	
Notes:	marked as "unsafe" with regard to threadsafety as implementation is allowed to do ressource allocation of buffer in the context of this call.	

⌋

C.5.2.2.2.4 GetSa

[SWS_DM_00297] Definition of API function apext::diag::uds_transport::UdsMessage::GetSa

Upstream requirements: RS_Diag_04147, RS_Diag_04174

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual Address GetSa () const noexcept;`	
Return value:	Address	The source address of the uds message.
Exception Safety:	exception safe	
Thread Safety:	reentrant	
Description:	Get the source address of the uds message.	

⌋

C.5.2.2.2.5 GetTa

[SWS_DM_00298] Definition of API function apext::diag::uds_transport::UdsMessage::GetTa

Upstream requirements: RS_Diag_04147, RS_Diag_04174

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual Address GetTa () const noexcept;`	
Return value:	Address	The target address of the uds message.
Exception Safety:	exception safe	
Thread Safety:	reentrant	
Description:	Get the target address of the uds message.	

⌋

C.5.2.2.2.6 GetTaType

[SWS_DM_00299] Definition of API function apext::diag::uds_transport::UdsMessage::GetTaType

Upstream requirements: RS_Diag_04147, RS_Diag_04174

⌈

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	`class apext::diag::uds_transport::UdsMessage`	
Syntax:	`virtual TargetAddressType GetTaType () const noexcept;`	
Return value:	TargetAddressType	The target address type of the uds message.
Exception Safety:	exception safe	
Thread Safety:	reentrant	
Description:	Get the target address type (phys/func) of the uds message.	

⌋

C.5.2.3 Protected Member Functions

C.5.2.3.1 Special Member Functions

C.5.2.3.1.1 Default Constructor

[SWS_DM_09012] Definition of API function apext::diag::uds_transport::UdsMessage::UdsMessage

Upstream requirements: RS_Diag_04147

⌈

Kind:	function
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	`class apext::diag::uds_transport::UdsMessage`
Syntax:	`UdsMessage () noexcept;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	non public default ctor. The default ctor is protected as we want to forbid, that UdsTransport Protocol handlers do create UdsMessages on its own! Only DM is allowed to create and hands over UdsMessagePtrs to UdsTransportProtocolHandler.
Visibility:	protected

⌋

C.6 Sequence Diagrams of UDS Transport Layer Interaction

C.6.1 Lifecycle

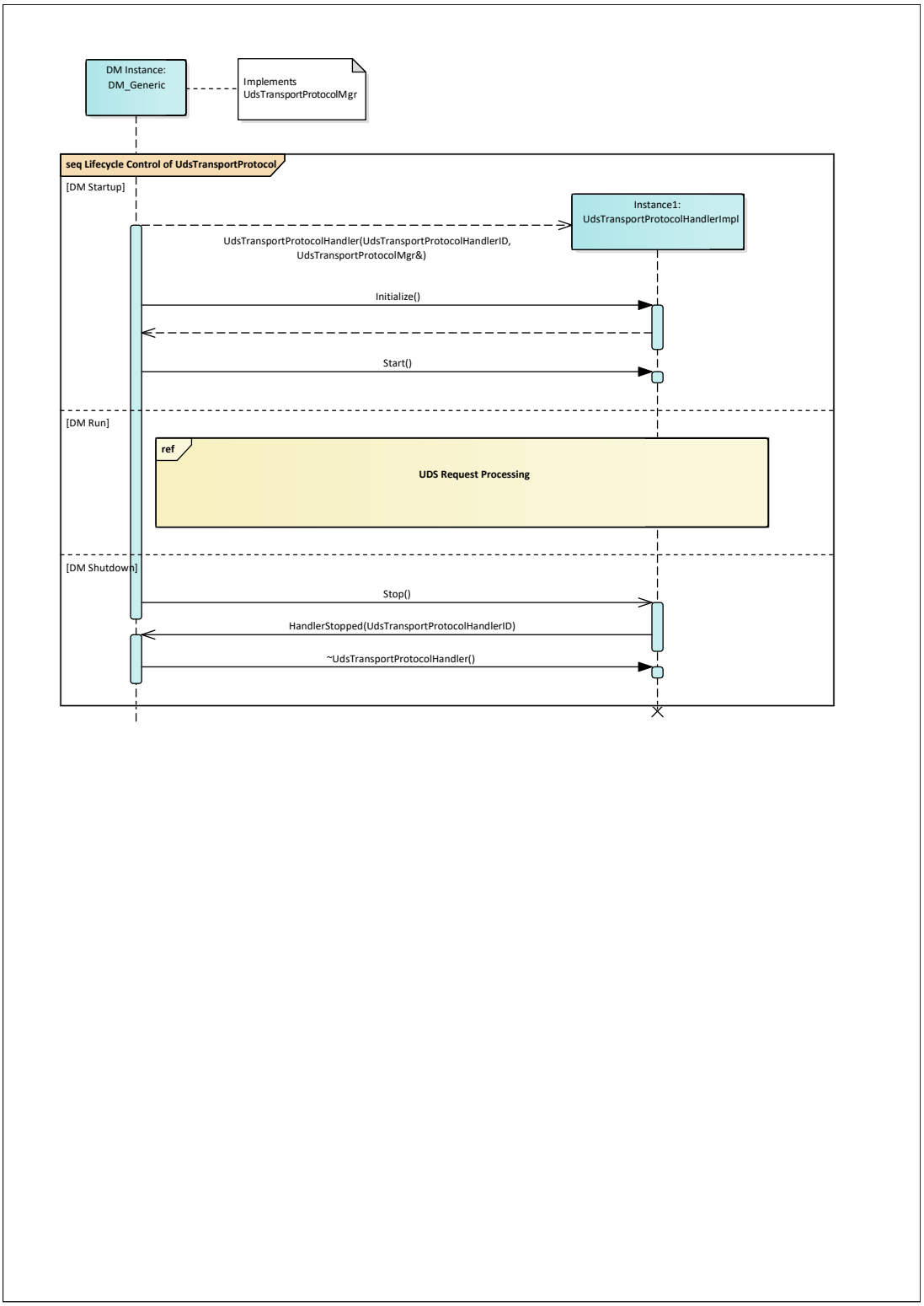

Figure C.1: UDS Transport Lifecycle

C.6.2 UDS Request Processing

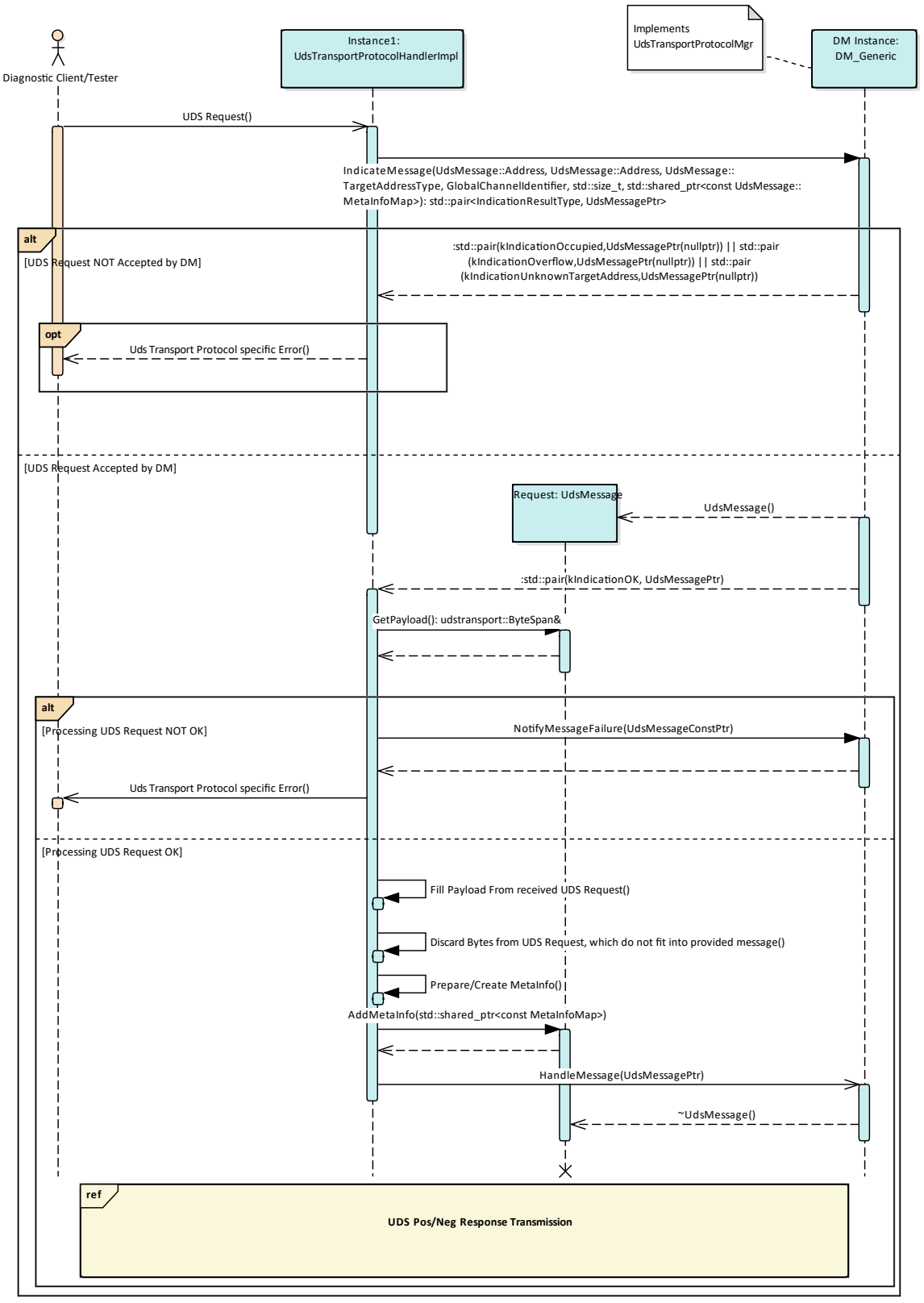

Figure C.2: UDS Transport Request Processing

C.6.3 UDS Response Transmission

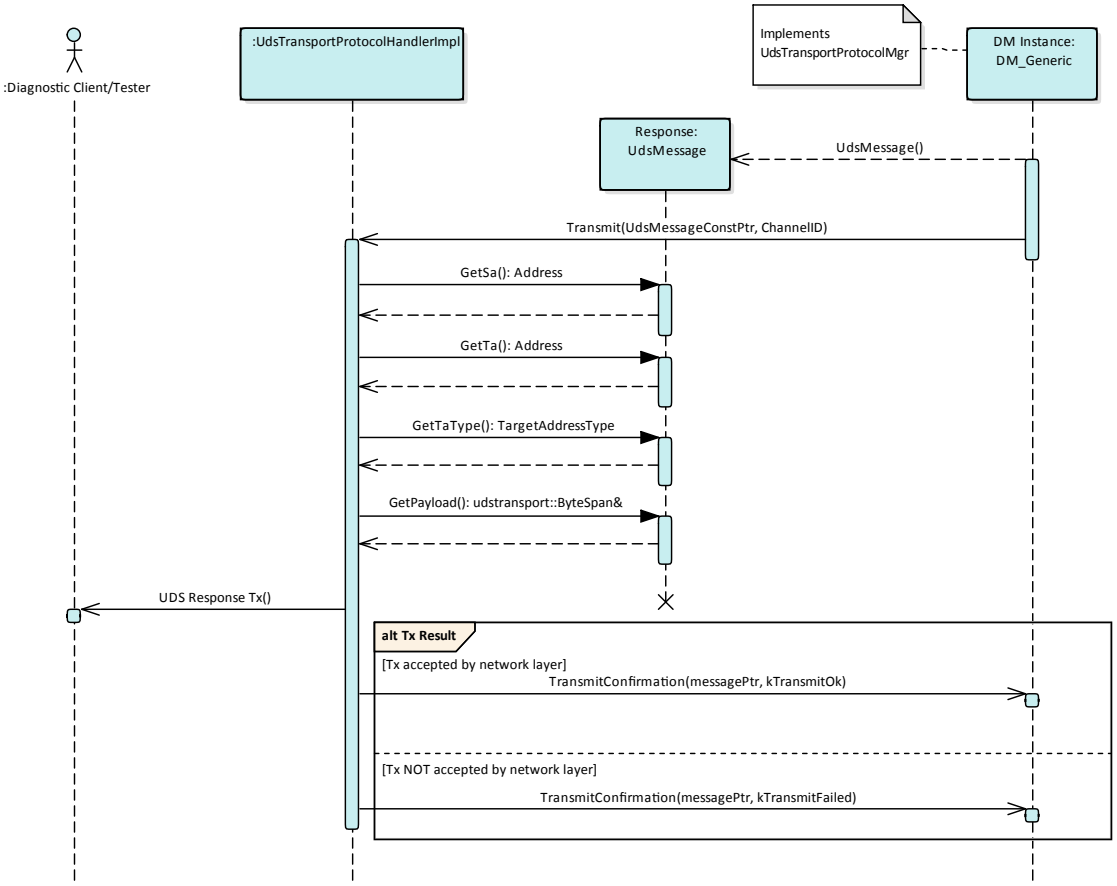

Figure C.3: UDS Response Transmission

C.6.4 Channel Reestablishment

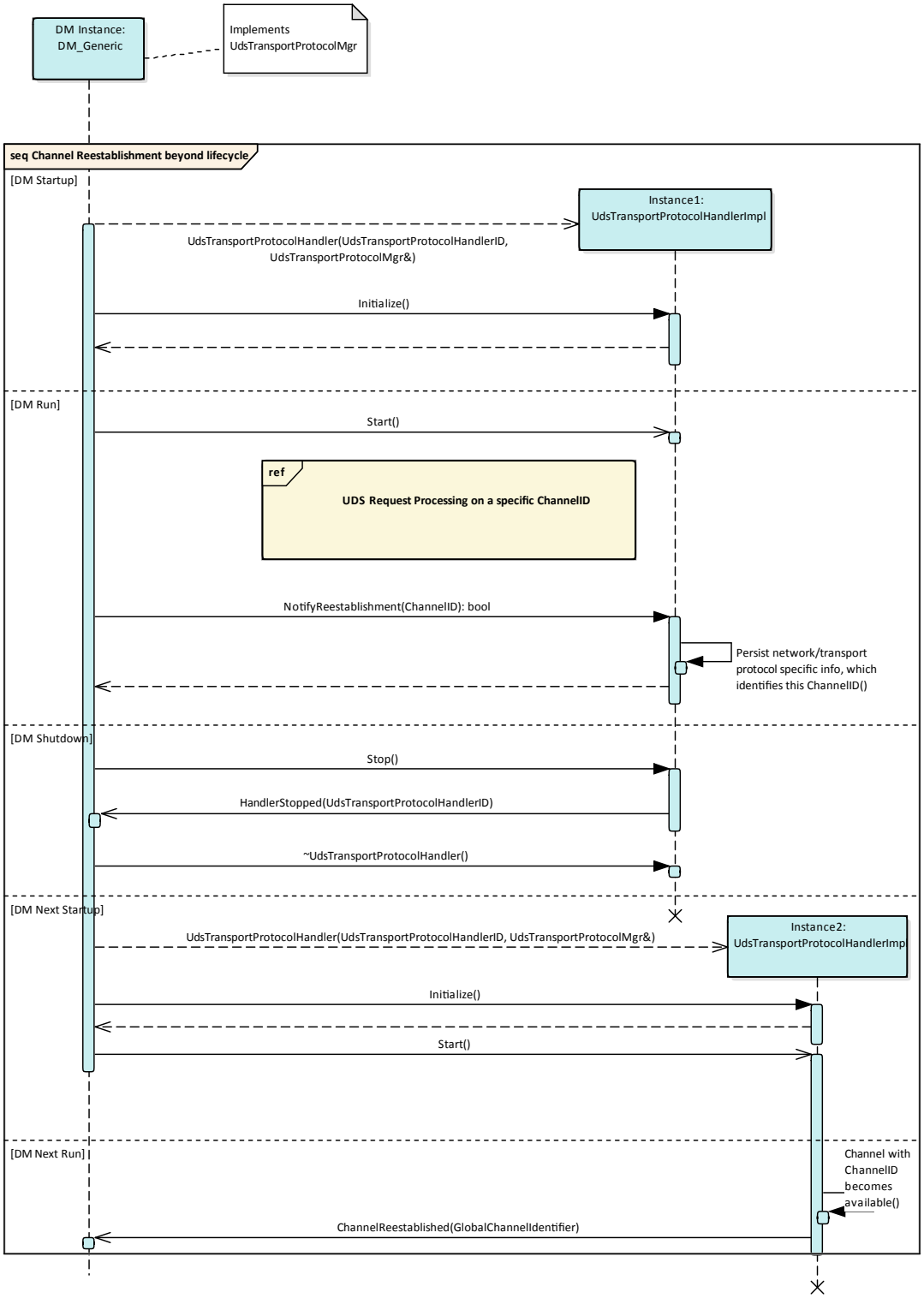

Figure C.4: UDS Transport Channel Reestablishment

D Not implemented requirements

This chapter lists all functional requirements specified in the corresponding requirement specifications that are not implemented or violated by this specification and provides a rationale.

D.1 Not applicable requirements

[SWS_DM_NA]

Status: DRAFT
Upstream requirements: RS_Diag_04171, RS_Diag_04195, RS_Diag_04218

⌈These requirements are not applicable as they are not within the scope of this release.⌋

E History of Constraints and Specification Items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

E.1 Constraint and Specification Item History of this document according to AUTOSAR Release 17-10

E.1.1 Added Specification Items in 17-10

Number	Heading
[SWS_DM_00277]	Cancellation of `Active Protocol` in case of External Service Processing
[SWS_DM_00278]	Cancellation of `Active Protocol` in case of Internal Processing
[SWS_DM_00279]	Cancellation of `Active Protocol` before Response Transmission
[SWS_DM_00280]	Cancellation of `Active Protocol` during Response Transmission
[SWS_DM_00281]	Cancellation of `Active Protocol` in Non-Default Session
[SWS_DM_00282]	Handling of CurrentActiveProtocols
[SWS_DM_00284]	SecurityAccess Service Interface
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00287]	Configurable environmental condition check criteria
[SWS_DM_00288]	Configurable environmental condition check evaluates to `TRUE`
[SWS_DM_00289]	Configurable environmental condition check evaluates to `FALSE`
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response
[SWS_DM_00291]	UdsMessage class
[SWS_DM_00292]	UdsMessage non public constructors
[SWS_DM_00293]	UdsMessage Address type
[SWS_DM_00294]	meta info map type
[SWS_DM_00295]	meta info map vendor type
[SWS_DM_00296]	TargetAddressType Address type
[SWS_DM_00297]	GetSa method
[SWS_DM_00298]	GetTa method
[SWS_DM_00299]	GetTaType method
[SWS_DM_00300]	GetPayload method readonly
[SWS_DM_00301]	GetPayload method
[SWS_DM_00302]	AddMetaInfo method
[SWS_DM_00303]	UdsMessage Pointer
[SWS_DM_00304]	Const UdsMessage Pointer

▽

△

Number	Heading
[SWS_DM_00305]	Const UdsMessage Pointer vendor type
[SWS_DM_00306]	UdsTransportProtocolMgr class
[SWS_DM_00307]	TransmissionResult type
[SWS_DM_00308]	Global Channel Identifier type
[SWS_DM_00309]	IndicateMessage method
[SWS_DM_00310]	NotifyMessageFailure method
[SWS_DM_00311]	HandleMessage method
[SWS_DM_00312]	TransmitConfirmation method
[SWS_DM_00313]	ChannelReestablished method
[SWS_DM_00314]	HandlerStopped method
[SWS_DM_00315]	UdsTransportProtocolHandler class
[SWS_DM_00316]	Header file
[SWS_DM_00317]	UdsTransportProtocolHandler constructor
[SWS_DM_00318]	UdsTransportProtocolHandler destructor
[SWS_DM_00319]	Initialize method
[SWS_DM_00320]	UdsTransportProtocolHandler UdsTransportProtocolMgr member
[SWS_DM_00321]	constructor member initialization
[SWS_DM_00322]	Start method
[SWS_DM_00323]	Stop method
[SWS_DM_00324]	UdsTransportProtocolHandler UdsTransportProtocolHandlerID member
[SWS_DM_00325]	GetHandlerID method
[SWS_DM_00326]	NotifyReestablishment method
[SWS_DM_00327]	Transmit method
[SWS_DM_00328]	UdsMessage Pointer vendor type
[SWS_DM_00329]	Lifecycle management of an `Uds Transport Protocol` implementation
[SWS_DM_00330]	Construction of an `Uds Transport Protocol` implementation
[SWS_DM_00331]	Initialization of an `Uds Transport Protocol` implementation
[SWS_DM_00332]	Starting of an `Uds Transport Protocol` implementation
[SWS_DM_00333]	Stopping of an `Uds Transport Protocol` implementation
[SWS_DM_00334]	UdsTransportProtocolMgr may be an abstract class
[SWS_DM_00335]	Header file
[SWS_DM_00336]	UdsTransportProtocolHandlerID
[SWS_DM_00337]	ChannelID
[SWS_DM_00338]	ByteVector
[SWS_DM_00339]	ByteVector vendor type
[SWS_DM_00340]	Waiting for Stop confirmation
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00343]	Acceptance of UDS message reception

▽

△

Number	Heading
[SWS_DM_00344]	Refusal of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00363]	Positive response processing
[SWS_DM_00364]	Negative response processing
[SWS_DM_00365]	Suppression of response
[SWS_DM_00366]	Suppression of response for functional requests
[SWS_DM_00367]	No service processing
[SWS_DM_00368]	Sending busy responses
[SWS_DM_00369]	Max. number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19

Table E.1: Added Specification Items in 17-10

E.1.2 Changed Specification Items in 17-10

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00004]	Operation cycle persistency

▽

△

Number	Heading
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00023]	Debounce counter jump down behavior
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00046]	Each `Diagnostic Protocol` has own session resources
[SWS_DM_00047]	Each `Diagnostic Protocol` has own security-level resources
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00049]	Refusal of second diagnostic request from different diagnostic client with BusyRepeatRequest
[SWS_DM_00051]	Cancellation of `Active Protocol` with lower priority
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00066]	Monitor initialization
[SWS_DM_00072]	Availability of enable condition service interfaces
[SWS_DM_00074]	Unsatisfied enable conditions
[SWS_DM_00088]	ControlDTCSetting influence
[SWS_DM_00089]	Reporting PREPASSED or PREFAILED for events without assigned debouncing algorithm
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00098]	UDS message checks
[SWS_DM_00099]	Supported Service SID level checks
[SWS_DM_00100]	Supported Service subfunction level checks
[SWS_DM_00101]	Session Access SID level Permission
[SWS_DM_00102]	Session Access subfunction level Permission
[SWS_DM_00103]	Security Access level Permission
[SWS_DM_00105]	Configurable Manufacturer Permission Check Services
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00107]	Configurable Supplier Permission Check Services
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00136]	Request upload service processing
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00166]	Trigger to process event status

▽

△

Number	Heading
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00172]	Reaction on Unsupported DataIdentifier
[SWS_DM_00176]	External ReadDataByIdentifier processing
[SWS_DM_00177]	Negative Response processing
[SWS_DM_00179]	Positive Response processing
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00188]	Reaction on Unsupported DataIdentifier
[SWS_DM_00189]	WriteDataByIdentifier processing
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00205]	Providing the `VIN` in DoIP protocol messages
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00252]	Reaction on Unsupported Subfunction
[SWS_DM_00258]	Cancellation of `Active Protocol` in non-default session
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00273]	Notification event upon `snapshot record` updates
[SWS_DM_00274]	Definition of an active `Diagnostic Protocol`

Table E.2: Changed Specification Items in 17-10

E.1.3 Deleted Specification Items in 17-10

Number	Heading
[SWS_DM_00001]	Availability of operation cycle service interfaces
[SWS_DM_00053]	Cancellation of `Active Protocol`
[SWS_DM_00054]	Generic UDS Service Interface

▽

△

Number	Heading
[SWS_DM_00073]	Checking enable conditions after status reports
[SWS_DM_00075]	Fulfilled enable conditions
[SWS_DM_00076]	Checking storage conditions in case the storage of event-related data is triggered
[SWS_DM_00077]	Checking storage conditions in case the update of event-related data is triggered
[SWS_DM_00081]	Routine Service Interface
[SWS_DM_00093]	Service Validation Interface
[SWS_DM_00094]	Data Services Interface
[SWS_DM_00149]	DTC related data
[SWS_DM_00157]	Snapshot record record data layout
[SWS_DM_00171]	Check for Supported DataIdentifier
[SWS_DM_00187]	Check for Supported DataIdentifier
[SWS_DM_00204]	Reaction on Unsupported Subfunction
[SWS_DM_00251]	Check for Supported Subfunction
[SWS_DM_CONSTR_00275]	Response processing after the actual reset

Table E.3: Deleted Specification Items in 17-10

E.1.4 Added Constraints in 17-10

none

E.1.5 Changed Constraints in 17-10

none

E.1.6 Deleted Constraints in 17-10

none

E.2 Constraint and Specification Item History of this document according to AUTOSAR Release 18-03

E.2.1 Added Specification Items in 18-03

Number	Heading
[SWS_DM_00001]	SRS Diagnostics
[SWS_DM_00376]	Positive response processing
[SWS_DM_00377]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00378]	ControlDTCSetting influence (reset)
[SWS_DM_00379]	Handling of storage conditions
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00382]	Session timeout start
[SWS_DM_00383]	Session timeout stop
[SWS_DM_00384]	IndicationResult type
[SWS_DM_00385]	Acceptance of UDS message reception
[SWS_DM_00386]	Ignoring UDS message reception because DM is busy
[SWS_DM_00387]	Ignoring UDS message reception because DM has no (memory) ressources
[SWS_DM_00388]	Filling provided UdsMessage
[SWS_DM_00389]	Skipping Forwarding of UDS message
[SWS_DM_00390]	Dispatching physical Request
[SWS_DM_00391]	Dispatching functional Request
[SWS_DM_00392]	Properties of returned UdsMessage
[SWS_DM_00393]	Retrieving data for `internal DiagnosticDataElement`s
[SWS_DM_00397]	Retrieving data for `external DiagnosticDataElement`s
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00402]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00403]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00404]	Default Service Interface for reading `DiagnosticDataIdentifier`
[SWS_DM_00405]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00406]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00407]	Default Service Interface for writing `DiagnosticDataIdentifier`
[SWS_DM_00408]	Retrieving data for requested DataIdentifier
[SWS_DM_00409]	Check supported DataIdentifier
[SWS_DM_00410]	Check session permission
[SWS_DM_00411]	Check security level permission
[SWS_DM_00412]	Check requested number of DataIdentifiers

▽

△

Number	Heading
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00415]	Check supported DataIdentifier
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00434]	Providing the PowerMode in DoIP protocol messages
[SWS_DM_CONSTR_00394]	Internal DiagnosticDataElements are read-only
[SWS_DM_CONSTR_00395]	Restriction on DEM-exclusive DiagnosticDataElements
[SWS_DM_CONSTR_00396]	Restriction on DCM-exclusive DiagnosticDataElements

Table E.4: Added Specification Items in 18-03

E.2.2 Changed Specification Items in 18-03

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00005]	DoIP Support
[SWS_DM_00007]	Uniqueness of diagnostic events
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00013]	Events without debouncing
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00017]	Calculation of the FDC based on the internal debounce counter
[SWS_DM_00018]	Internal debounce counter init and storage
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00022]	Debounce counter jump up behavior
[SWS_DM_00023]	Debounce counter jump down behavior
[SWS_DM_00024]	Qualified failed event using counter-based debouncing

▽

△

Number	Heading
[SWS_DM_00025]	Qualified passed event using counter-based debouncing
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00028]	Debounce counter persistency
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00033]	Debounce timer behavior upon reported failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00036]	Debounce timer behavior upon reported passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00039]	Resetting the debounce counter upon starting or restarting an operation cycle
[SWS_DM_00040]	Definition of debounce counter reset
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00042]	Cancelling external service processors
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00046]	Each `Diagnostic Protocol` has own session resources
[SWS_DM_00047]	Each `Diagnostic Protocol` has own security-level resources
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00049]	Refusal of second diagnostic request from different diagnostic client with BusyRepeatRequest
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00055]	Supported event memories
[SWS_DM_00057]	Availability of a user-defined event memory
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00060]	Set of supported DTCs
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[17] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord
[SWS_DM_00064]	Definition of DTC groups
[SWS_DM_00065]	Always supported availability of the group of all DTCs
[SWS_DM_00069]	Monitor initialization for enable condition reenabling reason

▽

△

Number	Heading
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00085]	Internal debounce counter init
[SWS_DM_00086]	Resetting the debounce counter after clearing DTC
[SWS_DM_00087]	Enable condition influence on debouncing behavior (freeze)
[SWS_DM_00088]	ControlDTCSetting influence (freeze)
[SWS_DM_00089]	Reporting PREPASSED or PREFAILED for events without assigned debouncing algorithm
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00091]	Evaluation of ClearDiagnosticInformation parameters
[SWS_DM_00092]	Parameter range check for groupOfDTC request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00097]	Abort on failed verification step
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service 0x14
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00116]	Clearing a DTC group
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00118]	Event specific configuration to allow clearing of a DTC
[SWS_DM_00119]	Init value for events with clear allowed information
[SWS_DM_00120]	Description of application interface to control the clear event behavior
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block status byte clearing during a clear DTC operation
[SWS_DM_00124]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[SWS_DM_00128]	Realisation of UDS service 0x34 RequestDownload
[SWS_DM_00129]	Supported addressAndLengthFormatIdentifier
[SWS_DM_00130]	Not supported addressAndLengthFormatIdentifier
[SWS_DM_00136]	Request upload service processing
[SWS_DM_00138]	Transfer data service processing
[SWS_DM_00139]	Transfer data service validation
[SWS_DM_00142]	Transfer data service processing
[SWS_DM_00143]	Transfer data service validation
[SWS_DM_00144]	Parallel clearing DTCs in different DiagnosticMemoryDestination

▽

△

Number	Heading
[SWS_DM_00145]	Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination
[SWS_DM_00146]	Unlock clear DTC operation for one DiagnosticMemoryDestination
[SWS_DM_00147]	Behavior while trying to clear DTCs on a locked DiagnosticMemoryDestination
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00151]	Snapshot record numeration
[SWS_DM_00152]	Number of snapshot records for a DTC
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00154]	Number of extended data for a DTC
[SWS_DM_00155]	Extended data record numeration
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00159]	Allow only to clear GroupOfAllDTCs
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00161]	Negative response on not supported GroupOfDTC parameter
[SWS_DM_00162]	Point in time for positive response for ClearDTC
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of DiagnosticMonitor service interfaces
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnosticInformation
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00205]	Providing the VIN in DoIP protocol messages
[SWS_DM_00213]	DTC status processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00217]	DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	Confirmation
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about the changes of the status byte
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00229]	Support of UDS service ControlDTCSetting

▽

△

Number	Heading
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00238]	Aging and healing
[SWS_DM_00239]	Aging counter
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Reoccurrence after aging
[SWS_DM_00243]	Aging-related UDS status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00258]	Cancellation of `Active Protocol` in non-default session
[SWS_DM_00259]	Completion of already `Active Protocol`s in default session
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00262]	Common semantic behavior for ClearDTC triggered via diagnostics or application
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00273]	Notification event upon `snapshot record` updates
[SWS_DM_00277]	Cancellation of `Active Protocol` in case of External Service Processing
[SWS_DM_00278]	Cancellation of `Active Protocol` in case of Internal Processing
[SWS_DM_00279]	Cancellation of `Active Protocol` before Response Transmission
[SWS_DM_00280]	Cancellation of `Active Protocol` at Response Transmission
[SWS_DM_00281]	Cancellation of active DiagnosticConversation in Non-Default Session
[SWS_DM_00282]	Handling of non-/active diagnostic conversations
[SWS_DM_00286]	Configurable environmental condition check execution

▽

△

Number	Heading
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response
[SWS_DM_00309]	IndicateMessage method
[SWS_DM_00316]	Header file
[SWS_DM_00329]	Lifecycle management of an `Uds Transport Protocol` implementation
[SWS_DM_00330]	Construction of an `Uds Transport Protocol` implementation
[SWS_DM_00331]	Initialization of an `Uds Transport Protocol` implementation
[SWS_DM_00332]	Starting of an `Uds Transport Protocol` implementation
[SWS_DM_00333]	Stopping of an `Uds Transport Protocol` implementation
[SWS_DM_00335]	Header file
[SWS_DM_00340]	Waiting for Stop confirmation
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00363]	Unsupported Subfunction
[SWS_DM_00366]	Suppression of response for functional requests
[SWS_DM_00369]	Max. number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_CONSTR_00059]	Restriction on supported DTC format
[SWS_DM_CONSTR_00082]	Restriction on the configuration of the DTC group GroupOfAllDTCs
[SWS_DM_CONSTR_00084]	Each DTC shall be assigned to an event memory destination

▽

△

Number	Heading
[SWS_DM_CONSTR_00168]	Required operation cycles for diagnostic events
[SWS_DM_CONSTR_00206]	Supported format for data identifier for VINDataIdentifier
[SWS_DM_CONSTR_00207]	Required VINDataIdentifier

Table E.5: Changed Specification Items in 18-03

E.2.3 Deleted Specification Items in 18-03

Number	Heading
[SWS_DM_00072]	Availability of enable condition service interfaces
[SWS_DM_00078]	Unsatisfied storage conditions
[SWS_DM_00079]	Fulfilled storage conditions
[SWS_DM_00172]	Reaction on Unsupported DataIdentifier
[SWS_DM_00173]	Classification as Internally implemented DID
[SWS_DM_00174]	Internally implemented DID ActiveDiagnosticSessionDataIdentifier
[SWS_DM_00175]	Classification as Externally implemented DID
[SWS_DM_00176]	External ReadDataByIdentifier processing
[SWS_DM_00178]	Check requested number of DataIdentifiers
[SWS_DM_00179]	Positive Response processing
[SWS_DM_00188]	Reaction on Unsupported DataIdentifier
[SWS_DM_00189]	WriteDataByIdentifier processing
[SWS_DM_00190]	Negative Response processing
[SWS_DM_00191]	Positive Response processing
[SWS_DM_00264]	ClearDTC call on invalid DTCOrigin
[SWS_DM_00292]	UdsMessage non public constructors
[SWS_DM_00343]	Acceptance of UDS message reception
[SWS_DM_00344]	Refusal of UDS message reception

Table E.6: Deleted Specification Items in 18-03

E.2.4 Added Constraints in 18-03

none

E.2.5 Changed Constraints in 18-03

none

E.2.6 Deleted Constraints in 18-03

none

E.3 Constraint and Specification Item History of this document according to AUTOSAR Release 18-10

E.3.1 Added Specification Items in 18-10

Number	Heading
[SWS_DM_00421]	Identification of a Diagnostic Client
[SWS_DM_00422]	Instantiation of Diagnostic Conversation Service Interface
[SWS_DM_00423]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00424]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00426]	Assigning a UDS request to an existing Diagnostic Conversation
[SWS_DM_00427]	Priority of a Diagnostic Conversation
[SWS_DM_00428]	Treatment of priority values
[SWS_DM_00429]	Prioritization in case of Pseudo Parallel Mode and active non-default session
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[SWS_DM_00432]	Initial values for Diagnostic Conversation
[SWS_DM_00433]	Refusal of diagnostic request due to busy Diagnostic Conversation
[SWS_DM_00435]	Default session change trigger from AAs
[SWS_DM_00436]	Providing the GID in DoIP protocol messages
[SWS_DM_00437]	Check supported RoutineIdentifier on active security level
[SWS_DM_00438]	Check Support of UDS service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of UDS service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of UDS service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of UDS service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of UDS service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of UDS service RequestTransferExit (0x37) on active security level
[SWS_DM_00444]	Check Support of UDS service ControlDTCSetting (0x85) in active session

△

Number	Heading
[SWS_DM_00445]	Check Support of UDS service ControlDTCSetting (0x85) on active security level
[SWS_DM_00446]	Check Support of UDS service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of UDS service RequestDownload (0x34) on active security level
[SWS_DM_00448]	Check supported RoutineIdentifier in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00475]	DoIP Version
[SWS_DM_00476]	User Controlled Warning IndicatorRequest-bit
[SWS_DM_00477]	Not Storing of 'warningIndicatorRequested' bit
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00480]	Security Access Blocking Timer
[SWS_DM_00481]	Handling of DiagnosticClearConditions
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00483]	Cancellation trigger from AAs
[SWS_DM_00484]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00485]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00487]	Ignoring UDS message reception because of unknown target address
[SWS_DM_00491]	Realisation of UDS service 0x86 ResponseOnEvent
[SWS_DM_00492]	Client Server communication
[SWS_DM_00493]	Reestablishing of Client Server communication
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00495]	Start initialisation of ResponseOnEvent
[SWS_DM_00496]	Stop initialisation of ResponseOnEvent
[SWS_DM_00497]	Clear initialisation of ResponseOnEvent
[SWS_DM_00498]	Exclusive ResponseOnEvent ressources
[SWS_DM_00499]	Replacement of a not started ResponseOnEvent initialisation
[SWS_DM_00500]	Replacement of a started ResponseOnEvent initialisation
[SWS_DM_00501]	Behavior while trying ResponseOnEvent activation while ResponseOnEvent is not initialised
[SWS_DM_00503]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00504]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00507]	Length check on UDS Service 0x27 request with Subfunction for RequestSeed

▽

△

Number	Heading
[SWS_DM_00508]	Reading DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00509]	Writing DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00651]	NumberOfStoredEntries
[SWS_DM_09010]	
[SWS_DM_09012]	
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_09017]	
[SWS_DM_09021]	
[SWS_DM_09028]	
[SWS_DM_CONSTR_00208]	Delay time value for sharedTimer
[SWS_DM_NA]	

Table E.7: Added Specification Items in 18-10

E.3.2 Changed Specification Items in 18-10

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00004]	Operation cycle persistency
[SWS_DM_00005]	DoIP Support
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00011]	Selectability of parallelism mode
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00016]	Configurable number of supported parallel Diagnostic Conversations
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00046]	Each Diagnostic Conversation has its own session resources
[SWS_DM_00047]	Each Diagnostic Conversation has its own security-level resources
[SWS_DM_00049]	Refusal of diagnostic request due to prioritization with BusyRepeatRequest

▽

△

Number	Heading
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[17] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00089]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00091]	Evaluation of ClearDiagnosticInformation parameters
[SWS_DM_00092]	Parameter range check for groupOfDTC request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00098]	UDS message checks
[SWS_DM_00099]	Supported Service SID level checks
[SWS_DM_00100]	Supported Service subfunction level checks
[SWS_DM_00101]	Session Access SID level Permission
[SWS_DM_00102]	Session Access subfunction level Permission
[SWS_DM_00103]	Security Access level Permission
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service 0x14
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block status byte clearing during a clear DTC operation
[SWS_DM_00124]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00126]	Realisation of UDS service 0x3E TesterPresent
[SWS_DM_00127]	Availability of diagnostic service processors

▽

△

Number	Heading
[SWS_DM_00128]	Realization of UDS service RequestDownload (0x34)
[SWS_DM_00129]	Supported addressAndLengthFormatIdentifier
[SWS_DM_00130]	Not supported addressAndLengthFormatIdentifier
[SWS_DM_00131]	UDS service RequestDownload (0x34) processing
[SWS_DM_00134]	Realization of UDS service RequestUpload (0x35)
[SWS_DM_00136]	UDS service RequestUpload (0x35) processing
[SWS_DM_00137]	Realization of UDS service TransferData (0x36)
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00140]	Realisation of UDS service 0x28 CommunicationControl
[SWS_DM_00141]	Realization of UDS service RequestTransferExit (0x37)
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00159]	Allow only to clear `GroupOfAllDTCs`
[SWS_DM_00160]	Allow to clear single `DTCs`
[SWS_DM_00162]	Point in time for positive response for ClearDTC
[SWS_DM_00163]	Definition of a failed clear operation with event clear allowed and event combination
[SWS_DM_00164]	Definition of a failed clear operation with event clear allowed and clearing a group of DTCs
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of `DiagnosticMonitor` service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00170]	Realisation of UDS service ReadDataByIdentifier (0x22)
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00186]	Realisation of UDS service WriteDataByIdentifier (0x2E)
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00193]	Support of a user-defined fault memory clear request
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnosticInformation
[SWS_DM_00195]	Clearing a user-defined memory
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00199]	Positive Response processing
[SWS_DM_00201]	Realization of UDS service RoutineControl (0x31)
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction

▽

△

Number	Heading
[SWS_DM_00205]	Providing the VIN in DoIP protocol messages
[SWS_DM_00208]	Validation of the requested user-defined memory number
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00213]	DTC status processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00216]	DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00217]	DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	Confirmation
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about DTC status changes
[SWS_DM_00222]	Observability of indicator status
[SWS_DM_00226]	Support of UDS service DiagnosticSessionControl
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00228]	Switch to requested Diagnostic Session
[SWS_DM_00229]	Support of UDS service ControlDTCSetting (0x85)
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Re-occurrence after aging
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00252]	Reaction on Unsupported Subfunction

▽

△

Number	Heading
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00263]	ClearDTC call on invalid DTC or DTCgroup
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00266]	ClearDTC processing in case of memory errors
[SWS_DM_00267]	Possible failure of ClearDTC
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00273]	Notification event upon `snapshot record` updates
[SWS_DM_00277]	Cancellation of a Diagnostic Conversation in case of External Service Processing
[SWS_DM_00278]	Cancellation of a Diagnostic Conversation in case of Internal Processing
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00288]	Configurable environmental condition check evaluates to `TRUE`
[SWS_DM_00289]	Configurable environmental condition check evaluates to `FALSE`
[SWS_DM_00290]	Refusal of diagnostic request due to prioritization without response
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	
[SWS_DM_00298]	
[SWS_DM_00299]	
[SWS_DM_00300]	
[SWS_DM_00301]	
[SWS_DM_00302]	
[SWS_DM_00303]	
[SWS_DM_00304]	
[SWS_DM_00306]	
[SWS_DM_00307]	
[SWS_DM_00309]	
[SWS_DM_00310]	
[SWS_DM_00311]	
[SWS_DM_00312]	
[SWS_DM_00313]	

▽

△

Number	Heading
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	
[SWS_DM_00326]	
[SWS_DM_00327]	
[SWS_DM_00329]	Lifecycle management of an `Uds Transport Protocol` implementation
[SWS_DM_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00364]	Negative response processing
[SWS_DM_00365]	Suppression of positive response in accordance to ISO 14229-1[17]
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[17]
[SWS_DM_00367]	No service processing
[SWS_DM_00368]	Sending busy responses
[SWS_DM_00369]	Maximum number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_00376]	Positive response processing
[SWS_DM_00379]	Handling of storage conditions
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00384]	
[SWS_DM_00385]	Acceptance of UDS message reception
[SWS_DM_00386]	Ignoring UDS message reception because DM is busy
[SWS_DM_00393]	Retrieving data for `internal DiagnosticDataElement`s
[SWS_DM_00397]	Retrieving data for `external DiagnosticDataElement`s
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00404]	Default Service Interface for reading `DiagnosticDataIdentifier`

▽

△

Number	Heading
[SWS_DM_00407]	Default Service Interface for writing `DiagnosticDataIdentifier`
[SWS_DM_00408]	Retrieving data for requested DataIdentifier
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00434]	Providing the `PowerMode` in DoIP protocol messages

Table E.8: Changed Specification Items in 18-10

E.3.3 Deleted Specification Items in 18-10

Number	Heading
[SWS_DM_00001]	SRS Diagnostics
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00051]	Cancellation of Active Protocol with lower priority
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00066]	Monitor initialization
[SWS_DM_00105]	Configurable Manufacturer Permission Check Services
[SWS_DM_00107]	Configurable Supplier Permission Check Services
[SWS_DM_00118]	Event specific configuration to allow clearing of a DTC
[SWS_DM_00119]	Init value for events with clear allowed information
[SWS_DM_00120]	Description of application interface to control the clear event behavior
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[SWS_DM_00161]	Negative response on not supported GroupOfDTC parameter
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment

▽

△

Number	Heading
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00185]	No Match
[SWS_DM_00258]	Cancellation of Active Protocol in non-default session
[SWS_DM_00259]	Completion of already Active Protocols in default session
[SWS_DM_00274]	Definition of an active `Diagnostic Protocol`
[SWS_DM_00281]	Cancellation of active DiagnosticConversation in Non-Default Session
[SWS_DM_00282]	Handling of non-/active diagnostic conversations
[SWS_DM_00295]	meta info map vendor type
[SWS_DM_00305]	Const UdsMessage Pointer vendor type
[SWS_DM_00308]	Global Channel Identifier type
[SWS_DM_00316]	Header file
[SWS_DM_00317]	UdsTransportProtocolHandler constructor
[SWS_DM_00318]	UdsTransportProtocolHandler destructor
[SWS_DM_00320]	UdsTransportProtocolHandler UdsTransportProtocolMgr member
[SWS_DM_00321]	constructor member initialization
[SWS_DM_00324]	UdsTransportProtocolHandler UdsTransportProtocolHandlerID member
[SWS_DM_00328]	UdsMessage Pointer vendor type
[SWS_DM_00334]	UdsTransportProtocolMgr may be an abstract class
[SWS_DM_00335]	Header file
[SWS_DM_00339]	ByteVector vendor type
[SWS_DM_00402]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00403]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00405]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00406]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00410]	Check session permission
[SWS_DM_00411]	Check security level permission
[SWS_DM_CONSTR_00207]	Required VINDataIdentifier

Table E.9: Deleted Specification Items in 18-10

E.3.4 Added Constraints in 18-10

none

E.3.5 Changed Constraints in 18-10

none

E.3.6 Deleted Constraints in 18-10

none

E.4 Constraint and Specification Item History of this document according to AUTOSAR Release 19-03

E.4.1 Added Specification Items in 19-03

Number	Heading
[SWS_DM_00510]	Namespace of Service header files
[SWS_DM_00511]	Implementation Types header files existence
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00513]	Implementation Types header file namespace
[SWS_DM_00526]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00561]	Deployment of diagnostic PortInterfaces

▽

△

Number	Heading
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00566]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00569]	Handling of storage conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00600]	

▽

△

Number	Heading
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00617]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00620]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00644]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	

▽

△

Number	Heading
[SWS_DM_00658]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00691]	
[SWS_DM_00692]	
[SWS_DM_00693]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00710]	
[SWS_DM_00711]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00731]	
[SWS_DM_00732]	

▽

△

Number	Heading
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00750]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00754]	
[SWS_DM_00755]	
[SWS_DM_00756]	
[SWS_DM_00781]	NumberOfStoredEntries
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00784]	
[SWS_DM_00785]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00793]	
[SWS_DM_00794]	
[SWS_DM_00795]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	

▽

△

Number	Heading
[SWS_DM_00803]	

Table E.10: Added Specification Items in 19-03

E.4.2 Changed Specification Items in 19-03

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00064]	Definition of DTC groups
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00199]	Positive Response processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00216]	DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00218]	Trip Counter
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00296]	
[SWS_DM_00307]	
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00364]	Negative response processing
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[17]
[SWS_DM_00367]	No service processing
[SWS_DM_00376]	Positive response processing

▽

△

Number	Heading
[SWS_DM_00382]	Session timeout start
[SWS_DM_00383]	Session timeout stop
[SWS_DM_00384]	
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00436]	Providing the GID in DoIP protocol messages
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00503]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00504]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00651]	
[SWS_DM_09017]	
[SWS_DM_CONSTR_00395]	Restriction on DEM-exclusive DiagnosticDataElements

Table E.11: Changed Specification Items in 19-03

E.4.3 Deleted Specification Items in 19-03

Number	Heading
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00483]	Cancellation trigger from AAs
[SWS_DM_09028]	

Table E.12: Deleted Specification Items in 19-03

E.4.4 Added Constraints in 19-03

none

E.4.5 Changed Constraints in 19-03

none

E.4.6 Deleted Constraints in 19-03

none

E.5 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11

E.5.1 Added Specification Items in 19-11

Number	Heading
[SWS_DM_00450]	Security Access subfunction level Permission
[SWS_DM_00502]	Support for Custom Diagnostic Services
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00645]	
[SWS_DM_00659]	
[SWS_DM_00660]	
[SWS_DM_00661]	
[SWS_DM_00662]	
[SWS_DM_00690]	
[SWS_DM_00702]	
[SWS_DM_00730]	
[SWS_DM_00760]	
[SWS_DM_00761]	
[SWS_DM_00762]	
[SWS_DM_00763]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00767]	
[SWS_DM_00770]	
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00804]	
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	

▽

△

Number	Heading
[SWS_DM_00809]	
[SWS_DM_00810]	
[SWS_DM_00811]	Re-enabling of ControlDTCSetting by Diagnostic Application
[SWS_DM_00812]	Re-enabling on transition to default session
[SWS_DM_00813]	Providing the GID in DoIP protocol messages
[SWS_DM_00814]	Providing the PowerMode in DoIP protocol messages
[SWS_DM_00815]	When to send Vehicle announcement messages on interfaces without activation line control
[SWS_DM_00816]	Notification of activation line status change on activation line controlled network interfaces
[SWS_DM_00820]	
[SWS_DM_00821]	
[SWS_DM_00822]	
[SWS_DM_00830]	
[SWS_DM_00831]	
[SWS_DM_00832]	
[SWS_DM_00833]	
[SWS_DM_00834]	
[SWS_DM_00835]	
[SWS_DM_00836]	
[SWS_DM_00837]	
[SWS_DM_00840]	Instantiation of Diagnostic Conversation Interface
[SWS_DM_00841]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00842]	Default session change trigger from AAs
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00844]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00845]	Notification about session change
[SWS_DM_00846]	Notification about security-level change
[SWS_DM_00847]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00850]	Default Service Interface for reading DiagnosticDataIdentifier
[SWS_DM_00855]	Providing the VIN in DoIP protocol messages
[SWS_DM_00856]	Initial values for Diagnostic Conversation
[SWS_DM_00857]	Signature of Manufacturer Permission Check Method
[SWS_DM_00858]	Signature of Supplier Permission Check Method
[SWS_DM_00859]	Confirmation of service processing
[SWS_DM_00860]	No service processing
[SWS_DM_00861]	Negative response processing

▽

△

Number	Heading
[SWS_DM_00862]	Suppression of negative response for functional requests in accordance to ISO 14229-1[17]
[SWS_DM_00863]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00864]	Checking Supported Subfunction for CompareKey
[SWS_DM_00865]	Communication control service processing
[SWS_DM_00866]	Negative Response processing
[SWS_DM_00867]	UDS service RequestDownload (0x34) processing
[SWS_DM_00868]	UDS service RequestUpload (0x35) processing
[SWS_DM_00869]	UDS service TransferData (0x36) processing
[SWS_DM_00870]	UDS service TransferData (0x36) validation
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00872]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00873]	Diagnostic event processing interface
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00877]	Starting time-based event debouncing for failed
[SWS_DM_00878]	Starting time-based event debouncing for passed
[SWS_DM_00879]	Application resetting the debounce counter
[SWS_DM_00880]	Debounce time freeze request
[SWS_DM_00881]	Enable condition influence on debouncing behavior (freeze)
[SWS_DM_00882]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00883]	UDS DTC status bit transitions triggered by test results
[SWS_DM_00884]	Resetting the status of the DTC
[SWS_DM_00885]	UDS DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00886]	Observability of the status byte
[SWS_DM_00887]	Notification about DTC status changes
[SWS_DM_00888]	Observability of indicator status
[SWS_DM_00889]	Automatic starting of operation cycles
[SWS_DM_00890]	Automatic ending of operation cycles
[SWS_DM_00891]	Restart of operation cycles
[SWS_DM_00892]	Operation cycles are only ended once
[SWS_DM_00893]	Triggering for snapshot record storage
[SWS_DM_00894]	Notification event upon snapshot record updates
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00896]	Handling of DiagnosticClearConditions
[SWS_DM_00897]	Usage of ClearDTC Interface

▽

△

Number	Heading
[SWS_DM_00898]	ClearDTC call on invalid DTC or DTC group
[SWS_DM_00899]	ClearDTC called while another clear operation is in progress
[SWS_DM_00900]	ClearDTC processing in case of memory errors
[SWS_DM_00901]	Possible failure of ClearDTC
[SWS_DM_00902]	NumberOfStoredEntries
[SWS_DM_00903]	Reading DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00904]	Writing DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00905]	Retrieving data for external DiagnosticDataElements
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00907]	Default Service Interface for writing DiagnosticDataIdentifier
[SWS_DM_00908]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00909]	Support of Subfunction 0x01 (ON)
[SWS_DM_00910]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00911]	Instances of DTCInformation interface
[SWS_DM_09011]	
[SWS_DM_09013]	
[SWS_DM_09014]	
[SWS_DM_09018]	

Table E.13: Added Specification Items in 19-11

E.5.2 Changed Specification Items in 19-11

Number	Heading
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00024]	Qualified failed event using counter-based debouncing
[SWS_DM_00025]	Qualified passed event using counter-based debouncing
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00033]	Debounce timer behavior upon reported failed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00036]	Debounce timer behavior upon reported passed
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00217]	UDS DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	Trip Counter
[SWS_DM_00242]	Re-occurrence after Aging

▽

△

Number	Heading
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00296]	
[SWS_DM_00307]	
[SWS_DM_00393]	Retrieving data for internal DiagnosticDataElements
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00421]	Identification of a Diagnostic Client
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00426]	Assigning a UDS request to an existing Diagnostic Conversation
[SWS_DM_00427]	Priority of a Diagnostic Conversation
[SWS_DM_00428]	Treatment of priority values
[SWS_DM_00429]	Prioritization in active non-default session
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[SWS_DM_00433]	Refusal of diagnostic request due to busy Diagnostic Conversation
[SWS_DM_00437]	Check supported RoutineIdentifier subfunction on active security level
[SWS_DM_00448]	Check supported RoutineIdentifier subfunction in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00475]	DoIP Version
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00507]	Length check on UDS Service 0x27 request with Subfunction for Request-Seed
[SWS_DM_00526]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	

▽

△

Number	Heading
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	

▽

△

Number	Heading
[SWS_DM_00616]	
[SWS_DM_00618]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00640]	
[SWS_DM_00644]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	
[SWS_DM_00658]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00692]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	

▽

△

Number	Heading
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00731]	
[SWS_DM_00732]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00750]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00754]	
[SWS_DM_00755]	
[SWS_DM_00756]	
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	

▽

△

Number	Heading
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_09012]	
[SWS_DM_09017]	

Table E.14: Changed Specification Items in 19-11

E.5.3 Deleted Specification Items in 19-11

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00011]	Selectability of parallelism mode
[SWS_DM_00016]	Configurable number of supported parallel Diagnostic Conversations
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00087]	Enable condition influence on debouncing behavior (freeze)

▽

△

Number	Heading
[SWS_DM_00089]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00131]	UDS service RequestDownload (0x34) processing
[SWS_DM_00136]	UDS service RequestUpload (0x35) processing
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of `DiagnosticMonitor` service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00205]	Providing the `VIN` in DoIP protocol messages
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the `DTC`
[SWS_DM_00216]	`DTC` status bit transitions triggered by operation cycle changes
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about DTC status changes
[SWS_DM_00222]	Observability of indicator status
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00263]	ClearDTC call on invalid DTC or DTCgroup
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00266]	ClearDTC processing in case of memory errors

▽

△

Number	Heading
[SWS_DM_00267]	Possible failure of ClearDTC
[SWS_DM_00273]	Notification event upon snapshot record updates
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00364]	Negative response processing
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[17]
[SWS_DM_00367]	No service processing
[SWS_DM_00377]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00379]	Handling of storage conditions
[SWS_DM_00397]	Retrieving data for external DiagnosticDataElements
[SWS_DM_00404]	Default Service Interface for reading DiagnosticDataIdentifier
[SWS_DM_00407]	Default Service Interface for writing DiagnosticDataIdentifier
[SWS_DM_00408]	Retrieving data for requested DataIdentifier
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00422]	Instantiation of Diagnostic Conversation Service Interface
[SWS_DM_00423]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00424]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00432]	Initial values for Diagnostic Conversation
[SWS_DM_00434]	Providing the PowerMode in DoIP protocol messages
[SWS_DM_00435]	Default session change trigger from AAs
[SWS_DM_00436]	Providing the GID in DoIP protocol messages
[SWS_DM_00476]	User Controlled Warning IndicatorRequest-bit
[SWS_DM_00477]	Not Storing of 'warningIndicatorRequested' bit
[SWS_DM_00481]	Handling of DiagnosticClearConditions
[SWS_DM_00484]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00485]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00503]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00504]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00508]	Reading DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00509]	Writing DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00566]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00569]	Handling of storage conditions
[SWS_DM_00781]	NumberOfStoredEntries

Table E.15: Deleted Specification Items in 19-11

E.5.4 Added Constraints in 19-11

none

E.5.5 Changed Constraints in 19-11

none

E.5.6 Deleted Constraints in 19-11

none

E.6 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11

E.6.1 Added Specification Items in R20-11

Number	Heading
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	
[SWS_DM_00521]	
[SWS_DM_00522]	
[SWS_DM_00523]	
[SWS_DM_00524]	
[SWS_DM_00525]	
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	

▽

△

Number	Heading
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	
[SWS_DM_00786]	
[SWS_DM_00916]	Priority values
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00920]	Configuration of the event memory size
[SWS_DM_00921]	Configuration of Error Memory Overflow Indication as extended data record
[SWS_DM_00922]	Persistent storage for event memory overflow information
[SWS_DM_00923]	Event memory overflow set condition
[SWS_DM_00924]	Event memory overflow reset condition
[SWS_DM_00925]	Event memory overflow notifier on occurence
[SWS_DM_00926]	Event memory overflow notifier on clear
[SWS_DM_00927]	Disabled displacement
[SWS_DM_00928]	Priority and occurrence based displacement
[SWS_DM_00929]	Displacement strategy "full"
[SWS_DM_00930]	Displacement operation
[SWS_DM_00932]	UDS DTC status bit 3 / 'ConfirmedDTC' after displacement
[SWS_DM_00933]	UDS DTC status bit 5 / 'testFailedSinceLastClear' after displacement
[SWS_DM_00934]	Condition for discarding the new event
[SWS_DM_00935]	

▽

△

Number	Heading
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00945]	Occurrence Counter initial value
[SWS_DM_00946]	Occurrence Counter increment strategy 'testFailed'-only
[SWS_DM_00947]	Occurrence Counter increment strategy 'confirmedDtcBit'
[SWS_DM_00948]	Occurrence Counter upper limit
[SWS_DM_00949]	Generation and usage of internal DiagnosticDataElements
[SWS_DM_00950]	Configuration of DTC priority as extended data record
[SWS_DM_00951]	Configuration of DTC "current FDC" as extended data record
[SWS_DM_00952]	Configuration of DTC "max. FDC since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. FDC current cycle" as extended data record
[SWS_DM_00954]	Configuration of DTC "occurrence counter" as extended data record
[SWS_DM_00955]	Configuration of DTC "aging counter up/down" as extended data record
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00957]	Configuration of DTC "aging counter down" as extended data record
[SWS_DM_00958]	Default value for DTC "aging counter up" if aging is not allowed
[SWS_DM_00959]	Default value for DTC "aging counter down" if aging is not allowed
[SWS_DM_00961]	Configuration of a DTCs significance as extended data record
[SWS_DM_00962]	Configuration of a DTCs Failed Operation Cycles as extended data record
[SWS_DM_00963]	Configuration of a DTCs failed operation Cycles Since First Failed as extended data record
[SWS_DM_00964]	Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record
[SWS_DM_00965]	Caching of monitor results
[SWS_DM_00966]	Reporting of DTCStatusAvailabilityMask
[SWS_DM_00967]	Support of UDS service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00970]	Behavior of failed data element retrieval
[SWS_DM_00971]	
[SWS_DM_00972]	
[SWS_DM_00973]	

▽

△

Number	Heading
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00981]	Conditions of status based reporting order
[SWS_DM_00982]	Reporting order direction
[SWS_DM_00983]	Processing of Custom Diagnostic Services
[SWS_DM_00984]	Return of cancellation status
[SWS_DM_00989]	
[SWS_DM_00990]	
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01008]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01015]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01018]	ECUReset `ara::diag::ResetRequestType` check

▽

△

Number	Heading
[SWS_DM_01019]	Custom `ara::diag::ResetRequestType` processing
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after `ara::diag::EcuResetRequest::RequestReset` called
[SWS_DM_01023]	Positive response before reset assurance
[SWS_DM_01024]	Event Status processing
[SWS_DM_01025]	`Event status` bit transitions triggered by test results
[SWS_DM_01026]	Resetting the status of an `Event`
[SWS_DM_01027]	`Event status` bit transitions triggered by `operation cycle` changes
[SWS_DM_01028]	`Event status` bit transitions triggered by ClearDiagnosticInformation `UDS service`
[SWS_DM_01029]	Notification about `Event status` changes
[SWS_DM_01030]	Observability of the `UDS DTC status byte`
[SWS_DM_01031]	Notification about UDS `DTC` status changes
[SWS_DM_01032]	Handling of 'WIR' bit without connected indicators
[SWS_DM_01033]	User controlled set of WIR-bit
[SWS_DM_01034]	User controlled reset of WIR-bit
[SWS_DM_01035]	User controlled WIR-bit handling and ControlDTCSetting
[SWS_DM_01037]	Behavior of not configured `DiagnosticEvent.confirmationThreshold`
[SWS_DM_01038]	Reading Diagnostic Data Identifier by `ara::diag::GenericDataIdentifier` interface
[SWS_DM_01039]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_01040]	Realization of `UDS` service ReadDataByPeriodicIdentifier(0x2A)
[SWS_DM_01041]	Check requested number of periodic DataIdentifiers
[SWS_DM_01042]	Minimum length check for ReadDataByPeriodicIdentifier
[SWS_DM_01043]	Check supported periodic DataIdentifier
[SWS_DM_01044]	Check Transmission Mode
[SWS_DM_01045]	Check Scheduler Availability
[SWS_DM_01046]	Check supported DataIdentifier in active session
[SWS_DM_01047]	Check supported DataIdentifier on active security level
[SWS_DM_01048]	Check DataIdentifier for environmental conditions
[SWS_DM_01049]	Checks Dynamically Defined `DIDs` in ReadDataByPeriodicIdentifier
[SWS_DM_01050]	Periodic `DID` length check
[SWS_DM_01051]	DM behavior on transmission Mode stopSending without periodicDataIdentifier in the request
[SWS_DM_01052]	DM behavior on transmission Mode stopSending with supported periodicDataIdentifier in the request
[SWS_DM_01053]	DM behavior on transmission Mode stopSending with not supported periodicDataIdentifier in the request

▽

△

Number	Heading
[SWS_DM_01054]	Starting to transmit PDIDs after positive response
[SWS_DM_01055]	Reaction on ApplicationError
[SWS_DM_01056]	Optional condition checks for sending periodic DIDs
[SWS_DM_01057]	Optional stopping PDIDs after session change
[SWS_DM_01058]	Optional stopping PDIDs after security level change
[SWS_DM_01059]	No periodic DIDs in default session
[SWS_DM_01060]	Support of Scheduler type 1
[SWS_DM_01061]	Trigger all scheduled PDIDs per scheduler
[SWS_DM_01062]	Transmission of all PDIDs on the periodic connection
[SWS_DM_01063]	Transmission error behavior
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_01070]	Support of UDS service 0x2C in Adaptive AUTOSAR DM
[SWS_DM_01071]	No persistency of defined DDIDs
[SWS_DM_01072]	Persistency of defined DDIDs
[SWS_DM_01073]	DM behavior for subfunction 'defineByIdentifier'
[SWS_DM_01074]	Only static DIDs as sourceDataIdentifier
[SWS_DM_01075]	Maximum number of sourceDataIdentifiers in the request
[SWS_DM_01076]	Clearing all configured DDIDs
[SWS_DM_01077]	Clearing individual configured DDIDs
[SWS_DM_01078]	Clear DDIDs on session change
[SWS_DM_01079]	Session check for DDID
[SWS_DM_01080]	Security level check for DDID
[SWS_DM_01081]	Session check for sourceDataIdentifier
[SWS_DM_01082]	Security level check for sourceDataIdentifier
[SWS_DM_01083]	Use of configured DID ports to get DDID data
[SWS_DM_CONSTR_00960]	No support for DEM_AGINGCTR_UPCNT_FIRST_ACTIVE

Table E.16: Added Specification Items in R20-11

E.6.2 Changed Specification Items in R20-11

Number	Heading
[SWS_DM_00017]	Calculation of the FDC based on the internal debounce counter
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00062]	Mapping between ISO 14229-1[17] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00123]	Block clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00124]	Limited clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00150]	Primary trigger for event memory entry storage
[SWS_DM_00213]	DTC status processing
[SWS_DM_00218]	UDS DTC status bit 'kConfirmedDTC'
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00237]	Aging
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00369]	Maximum number of busy responses
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00437]	Security Level check for RoutineIdentifier
[SWS_DM_00448]	Check supported RoutineIdentifier subfunction in active session
[SWS_DM_00502]	Support for Custom Diagnostic Services
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	

▽

△

Number	Heading
[SWS_DM_00557]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00585]	
[SWS_DM_00587]	
[SWS_DM_00589]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00618]	
[SWS_DM_00619]	

▽

△

Number	Heading
[SWS_DM_00634]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00640]	
[SWS_DM_00650]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00670]	
[SWS_DM_00673]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00699]	
[SWS_DM_00722]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00732]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00762]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00787]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00797]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_00806]	
[SWS_DM_00808]	
[SWS_DM_00809]	

▽

△

Number	Heading
[SWS_DM_00836]	
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00857]	Signature of Manufacturer Permission Check Method
[SWS_DM_00858]	Signature of Supplier Permission Check Method
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00883]	UDS DTC status bit transitions triggered by test results
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00898]	ClearDTC call on invalid DTC or DTC group
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00908]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00909]	Support of Subfunction 0x01 (ON)
[SWS_DM_00910]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00911]	Instances of DTCInformation interface
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_CONSTR_00059]	Restriction on supported DTC format
[SWS_DM_CONSTR_00396]	Restriction on DCM-exclusive DiagnosticDataElements

Table E.17: Changed Specification Items in R20-11

E.6.3 Deleted Specification Items in R20-11

Number	Heading
[SWS_DM_00013]	Events without debouncing
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00284]	SecurityAccess Service Interface
[SWS_DM_00702]	
[SWS_DM_00884]	Resetting the status of the DTC
[SWS_DM_00885]	UDS DTC status bit transitions triggered by operation cycle changes

▽

△

Number	Heading
[SWS_DM_00887]	Notification about DTC status changes

Table E.18: Deleted Specification Items in R20-11

E.6.4 Added Constraints in R20-11

none

E.6.5 Changed Constraints in R20-11

none

E.6.6 Deleted Constraints in R20-11

none

E.7 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11

E.7.1 Added Specification Items in R21-11

Number	Heading
[SWS_DM_01084]	Triggering for snapshot record storage (calculated)
[SWS_DM_01085]	Triggering for snapshot record storage (configured, without update)
[SWS_DM_01086]	Triggering for snapshot record storage (configured, with update)
[SWS_DM_01087]	Snapshot record layout
[SWS_DM_01088]	
[SWS_DM_01089]	
[SWS_DM_01090]	Calling ExecuteReset() if positive response shall be sent after reset
[SWS_DM_01091]	Maximum time frame to trigger ExecuteReset()
[SWS_DM_01092]	EnableRapidPowerShutdown Positive Response
[SWS_DM_01093]	Caching of conditions
[SWS_DM_01094]	Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On

▽

△

Number	Heading
[SWS_DM_01095]	Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off
[SWS_DM_01096]	UDS service TransferData (0x36) upload processing
[SWS_DM_01097]	UDS service RequestTransferExit (0x37) upload processing
[SWS_DM_01098]	Starting ResponseOnEvent in single and multiple client scenarios
[SWS_DM_01099]	Debouncing parameters from DEXT
[SWS_DM_01100]	Debouncing Algorithm not fitting
[SWS_DM_01101]	Debouncing parameters from Monitor Constructor
[SWS_DM_01102]	
[SWS_DM_01103]	Caching of RestartOperationCycle
[SWS_DM_01104]	Operation Cycle restart
[SWS_DM_01105]	Restart OperationCycle during the processing of previous call
[SWS_DM_01106]	Applicability of Event Combination
[SWS_DM_01107]	DTC Status Byte calculation
[SWS_DM_01108]	Clear all DTCs event with event combination
[SWS_DM_01109]	UDS DTC status update for combined DTCs
[SWS_DM_01110]	Callbacks for combined UDS DTC status change
[SWS_DM_01111]	Fault detection counter for combined events
[SWS_DM_01112]	Event memory entry for events with the combination on storage
[SWS_DM_01113]	Aging counter for combined events
[SWS_DM_01114]	Data storage for event combination on retrieval
[SWS_DM_01115]	Data reporting for event combination on retrieval
[SWS_DM_01116]	Reporting order of snapshot and extended data records
[SWS_DM_01117]	Support of eventWindowTime values for ResponseOnEvent
[SWS_DM_01118]	Support of DEXT parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate
[SWS_DM_01119]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentfierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength
[SWS_DM_01122]	Support of DEXT parameter DiagnosticResponseOnEvent-Class.maxNumberOfStoredDTCStatusChangedEvents
[SWS_DM_01123]	
[SWS_DM_01124]	
[SWS_DM_01125]	
[SWS_DM_01126]	
[SWS_DM_01127]	
[SWS_DM_01128]	

▽

△

Number	Heading
[SWS_DM_01129]	
[SWS_DM_01130]	
[SWS_DM_01131]	
[SWS_DM_01132]	
[SWS_DM_01133]	
[SWS_DM_01134]	
[SWS_DM_01136]	
[SWS_DM_01137]	
[SWS_DM_01138]	
[SWS_DM_01139]	
[SWS_DM_01140]	
[SWS_DM_01141]	
[SWS_DM_01142]	
[SWS_DM_01143]	
[SWS_DM_01144]	
[SWS_DM_01145]	
[SWS_DM_01146]	
[SWS_DM_01147]	
[SWS_DM_01148]	
[SWS_DM_01149]	
[SWS_DM_01150]	
[SWS_DM_01151]	
[SWS_DM_01152]	
[SWS_DM_01153]	
[SWS_DM_01154]	
[SWS_DM_01155]	
[SWS_DM_01156]	
[SWS_DM_01157]	
[SWS_DM_01158]	
[SWS_DM_01159]	
[SWS_DM_01160]	
[SWS_DM_01161]	
[SWS_DM_01162]	
[SWS_DM_01163]	
[SWS_DM_01164]	
[SWS_DM_01165]	
[SWS_DM_01166]	
[SWS_DM_01167]	
[SWS_DM_01168]	

▽

△

Number	Heading
[SWS_DM_01169]	
[SWS_DM_01170]	
[SWS_DM_01171]	
[SWS_DM_01172]	
[SWS_DM_01173]	
[SWS_DM_01174]	
[SWS_DM_01175]	
[SWS_DM_01176]	
[SWS_DM_01177]	
[SWS_DM_01178]	
[SWS_DM_01179]	
[SWS_DM_01180]	
[SWS_DM_01181]	
[SWS_DM_01182]	
[SWS_DM_01183]	
[SWS_DM_01184]	
[SWS_DM_01185]	
[SWS_DM_01186]	
[SWS_DM_01187]	
[SWS_DM_01188]	
[SWS_DM_01189]	
[SWS_DM_01190]	
[SWS_DM_01191]	
[SWS_DM_01192]	
[SWS_DM_01193]	
[SWS_DM_01194]	
[SWS_DM_01195]	
[SWS_DM_01196]	
[SWS_DM_01197]	
[SWS_DM_01198]	
[SWS_DM_01199]	
[SWS_DM_01200]	
[SWS_DM_01201]	
[SWS_DM_01202]	Get ClientAuthentication Instance
[SWS_DM_01203]	GetAll ClientAuthentication Instance
[SWS_DM_01204]	Default Authentication Role
[SWS_DM_01205]	Default Authentication State
[SWS_DM_01206]	Set AuthenticationRole
[SWS_DM_01207]	Get Authentication State

▽

△

Number	Heading
[SWS_DM_01208]	Authentication State Change Notifier
[SWS_DM_01209]	Temporarily change Default Roles
[SWS_DM_01210]	DeAuthenticate due to client inactivity
[SWS_DM_01211]	Transition to DeAuthenticated state on S3server timeout
[SWS_DM_01212]	Transition from Authenticated to DeAuthenticated State
[SWS_DM_01213]	Set DynamicAccessList
[SWS_DM_01214]	Default DynamicAccessList
[SWS_DM_01215]	Extend the DynamicAccessList
[SWS_DM_01216]	Revoke an authentication
[SWS_DM_01217]	Refresh timeouts
[SWS_DM_01218]	Building a new DynamicAccessList
[SWS_DM_01219]	Adding patterns to a DynamicAccessList
[SWS_DM_01220]	Adding wildcards to a DynamicAccessList
[SWS_DM_01221]	End patterns of a DynamicAccessList
[SWS_DM_01222]	Finalize a DynamicAccessList
[SWS_DM_01223]	Diagnostic service role verification
[SWS_DM_01224]	Diagnostic service dynamic access-rights verification
[SWS_DM_01225]	Response behavior of services without access rights
[SWS_DM_01226]	Support of UDS service authentication
[SWS_DM_01227]	Configuration of authentication types
[SWS_DM_01228]	Mandatory sub functions
[SWS_DM_01229]	Support for authentication per Diagnostic Conversation
[SWS_DM_01230]	Processing the verifyCertificateUnidirectional request
[SWS_DM_01231]	Handling Negative return values of `ara::diag::Authentication::VerifyCertificateUnidirectional` method
[SWS_DM_01232]	Handling unspecified negative return values of `ara::diag::Authentication::VerifyCertificateUnidirectional` method
[SWS_DM_01233]	Successful verification of verifyCertificateUnidirectional
[SWS_DM_01234]	Unexpected verifyCertificateUnidirectional from a different client
[SWS_DM_01235]	Processing the verifyCertificateBidirectional request
[SWS_DM_01236]	Handling Negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method
[SWS_DM_01237]	Handling unspecified negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method
[SWS_DM_01238]	Successful verification of verifyCertificateBidirectional
[SWS_DM_01239]	Unexpected verifyCertificateBidirectional from a different client
[SWS_DM_01240]	Processing the proofOfOwnership request
[SWS_DM_01241]	Handling Negative return values of `ara::diag::Authentication::VerifyOwnership` method

▽

△

Number	Heading
[SWS_DM_01242]	Handling unspecified negative return values of `ara::diag::Authentication::VerifyOwnership` method
[SWS_DM_01243]	Successful verification of Client proofOfOwnership
[SWS_DM_01244]	Processing the deAuthenticate request
[SWS_DM_01245]	Successful completion of deAuthenticate
[SWS_DM_01246]	Processing the authenticationConfiguration request
[SWS_DM_01247]	Validation of the transmitCertificate certificateEvaluationId
[SWS_DM_01248]	Processing the transmitCertificate request
[SWS_DM_01249]	Handling Negative return values of `ara::diag::Authentication::TransmitCertificate` method
[SWS_DM_01250]	Handling unspecified negative return values of `ara::diag::Authentication::TransmitCertificate` method
[SWS_DM_01251]	Successful verification of transmitCertificate
[SWS_DM_-CONSTR_00961]	Limits of priority values

Table E.19: Added Specification Items in R21-11

E.7.2 Changed Specification Items in R21-11

Number	Heading
[SWS_DM_00018]	Internal debounce counter init
[SWS_DM_00022]	Internal debounce counter jump up behavior
[SWS_DM_00023]	Internal debounce counter jump down behavior
[SWS_DM_00039]	Resetting the internal debounce counter upon restarting an operation cycle
[SWS_DM_00040]	Definition of internal debounce counter reset
[SWS_DM_00086]	Resetting the internal debounce counter after clearing `DTC`
[SWS_DM_00088]	ControlDTCSetting influence (freeze)
[SWS_DM_00163]	Definition of a inhibited clear operation on single DTC
[SWS_DM_00164]	Definition of a inhibited clear operation for a group of DTCs
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	
[SWS_DM_00298]	
[SWS_DM_00299]	

▽

△

Number	Heading
[SWS_DM_00300]	
[SWS_DM_00301]	
[SWS_DM_00302]	
[SWS_DM_00303]	
[SWS_DM_00304]	
[SWS_DM_00306]	
[SWS_DM_00307]	
[SWS_DM_00309]	
[SWS_DM_00310]	
[SWS_DM_00311]	
[SWS_DM_00312]	
[SWS_DM_00313]	
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	
[SWS_DM_00326]	
[SWS_DM_00327]	
[SWS_DM_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00378]	ControlDTCSetting influence (reset)
[SWS_DM_00384]	
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations in active default session
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00491]	Realisation of UDS service 0x86 ResponseOnEvent
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	

▽

△

Number	Heading
[SWS_DM_00521]	
[SWS_DM_00522]	
[SWS_DM_00523]	
[SWS_DM_00524]	
[SWS_DM_00525]	
[SWS_DM_00526]	
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	

▽

△

Number	Heading
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00568]	Handling of `enable conditions`
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00600]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	

▽

△

Number	Heading
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00617]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00620]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00644]	
[SWS_DM_00645]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	

▽

△

Number	Heading
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	
[SWS_DM_00658]	
[SWS_DM_00659]	
[SWS_DM_00660]	
[SWS_DM_00661]	
[SWS_DM_00662]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00690]	
[SWS_DM_00691]	
[SWS_DM_00692]	
[SWS_DM_00693]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	

▽

△

Number	Heading
[SWS_DM_00710]	
[SWS_DM_00711]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00730]	
[SWS_DM_00731]	
[SWS_DM_00732]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00755]	
[SWS_DM_00760]	
[SWS_DM_00761]	
[SWS_DM_00762]	
[SWS_DM_00763]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00767]	
[SWS_DM_00770]	

▽

△

Number	Heading
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00784]	
[SWS_DM_00785]	
[SWS_DM_00786]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00793]	
[SWS_DM_00794]	
[SWS_DM_00795]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_00803]	
[SWS_DM_00804]	
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	
[SWS_DM_00809]	
[SWS_DM_00810]	
[SWS_DM_00820]	
[SWS_DM_00821]	
[SWS_DM_00822]	
[SWS_DM_00830]	

▽

△

Number	Heading
[SWS_DM_00831]	
[SWS_DM_00832]	
[SWS_DM_00833]	
[SWS_DM_00834]	
[SWS_DM_00835]	
[SWS_DM_00836]	
[SWS_DM_00837]	
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_00855]	Providing the VIN in DoIP protocol messages
[SWS_DM_00869]	UDS service TransferData (0x36) download processing
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) download processing
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00879]	Application resetting the internal debounce counter
[SWS_DM_00882]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00902]	NumberOfStoredEntries
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00971]	
[SWS_DM_00972]	
[SWS_DM_00973]	
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	

▽

△

Number	Heading
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00989]	
[SWS_DM_00990]	
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01023]	Calling ExecuteReset() if positive response shall be sent before reset
[SWS_DM_01027]	`Event status` bit transitions triggered by `operation cycle` restarting
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_09010]	
[SWS_DM_09011]	

▽

△

Number	Heading
[SWS_DM_09012]	
[SWS_DM_09013]	
[SWS_DM_09014]	
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_09017]	
[SWS_DM_09018]	
[SWS_DM_09021]	

Table E.20: Changed Specification Items in R21-11

E.7.3 Deleted Specification Items in R21-11

Number	Heading
[SWS_DM_00004]	Operation cycle persistency
[SWS_DM_00028]	Debounce counter persistency
[SWS_DM_00165]	Considering only events referencing a DTC
[SWS_DM_00438]	Check Support of UDS service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of UDS service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of UDS service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of UDS service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of UDS service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of UDS service RequestTransferExit (0x37) on active security level
[SWS_DM_00446]	Check Support of UDS service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of UDS service RequestDownload (0x34) on active security level
[SWS_DM_00492]	Client Server communication
[SWS_DM_00495]	Start initialisation of ResponseOnEvent
[SWS_DM_00496]	Stop initialisation of ResponseOnEvent
[SWS_DM_00497]	Clear initialisation of ResponseOnEvent
[SWS_DM_00498]	Exclusive ResponseOnEvent ressources
[SWS_DM_00499]	Replacement of a not started ResponseOnEvent initialisation
[SWS_DM_00500]	Replacement of a started ResponseOnEvent initialisation
[SWS_DM_00501]	Behavior while trying ResponseOnEvent activation while ResponseOnEvent is not initialised
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason

▽

△

Number	Heading
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00750]	
[SWS_DM_00754]	
[SWS_DM_00756]	
[SWS_DM_00870]	UDS service TransferData (0x36) validation
[SWS_DM_00872]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00889]	Automatic starting of operation cycles
[SWS_DM_00890]	Automatic ending of operation cycles
[SWS_DM_00891]	Restart of operation cycles
[SWS_DM_00892]	Operation cycles are only ended once
[SWS_DM_00893]	Triggering for `snapshot record` storage
[SWS_DM_00903]	Reading DiagnosticDataIdentifier configured for representing `VIN`
[SWS_DM_00904]	Writing DiagnosticDataIdentifier configured for representing `VIN`
[SWS_DM_01008]	
[SWS_DM_01015]	

Table E.21: Deleted Specification Items in R21-11

E.7.4 Added Constraints in R21-11

none

E.7.5 Changed Constraints in R21-11

none

E.7.6 Deleted Constraints in R21-11

none

E.8 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11

E.8.1 Added Specification Items in R22-11

Number	Heading
[SWS_DM_01252]	Support of manufacturer service validations
[SWS_DM_01253]	Support of supplier service validations
[SWS_DM_01254]	Continue service processing after validation
[SWS_DM_01255]	`NRC` after failed service validation
[SWS_DM_01256]	Support of UDS service ReadDTCInformation, Subfunction 0x03
[SWS_DM_01257]	ResourceTemporarilyNotAvailable NRC handling
[SWS_DM_01258]	Response handling
[SWS_DM_01259]	Validation of Security Level Locked in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`
[SWS_DM_01260]	Validation of Invalid Security Level in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`
[SWS_DM_01261]	Validation of Invalid Session Level in `ara::diag::Conversation::GetDiagnosticSessionShortName`
[SWS_DM_01262]	No storage of RoE events
[SWS_DM_01263]	Storage of RoE events
[SWS_DM_01264]	`DiagnosticAging.threshold` reached
[SWS_DM_01265]	`Aging` requires tested cycles only
[SWS_DM_01266]	Warning Indicator Request Activation
[SWS_DM_01267]	Reporting `kFdcThresholdReached` for monitor internal debouncing
[SWS_DM_01268]	Value of FaultDetectionCounter in case of monitor internal debouncing
[SWS_DM_01269]	Requesting `DTC` number for events without `DTC`
[SWS_DM_01270]	UDS Service RoutineControl (0x31) startRoutine processing with typed interface
[SWS_DM_01271]	UDS Service RoutineControl (0x31) stopRoutine processing with typed interface
[SWS_DM_01272]	UDS Service RoutineControl (0x31) requestRoutineResults processing with typed interface
[SWS_DM_01273]	Namespace for typed UDS Service RoutineControl (0x31)
[SWS_DM_01274]	Namespace for typed DiagnosticDataIdentifier interface
[SWS_DM_01275]	Namespace for typed DiagnosticDataElements interface
[SWS_DM_01276]	Triggering for `snapshot record` storage (calculated, maxNumberFreezeFrameRecords = 1)
[SWS_DM_01277]	Triggering for `snapshot record` storage (calculated, maxNumberFreezeFrameRecords > 1)
[SWS_DM_01278]	

▽

△

Number	Heading
[SWS_DM_01279]	
[SWS_DM_01280]	
[SWS_DM_01281]	
[SWS_DM_01282]	
[SWS_DM_01283]	
[SWS_DM_01284]	
[SWS_DM_01285]	
[SWS_DM_01286]	
[SWS_DM_01292]	
[SWS_DM_01293]	
[SWS_DM_01294]	
[SWS_DM_01296]	Behavior of service ResponseOnEvent with subfunction onDTCStatusChange for suppressed DTCs
[SWS_DM_01297]	Behavior of DTCInformation::SetNumberOfStoredEntriesNotifier for suppressed DTCs
[SWS_DM_01298]	Behavior of DTCInformation::SetDTCStatusChangedNotifier for suppressed DTCs
[SWS_DM_01299]	Enabling of the notification ara::diag::DTCInformation::SetDTCStatusChangedNotifier for suppressed DTCs
[SWS_DM_01300]	Behavior of the ara::diag::DTCInformation class for suppressed DTCs
[SWS_DM_01301]	Behavior of ara::diag::Event class methods for suppressed DTCs
[SWS_DM_01302]	Behavior of Diagnostic Client services ClearDiagnosticInformation for suppressed DTCs inside a group
[SWS_DM_01303]	Behavior of Diagnostic Client services ClearDiagnosticInformation for suppressed DTCs
[SWS_DM_01304]	Behavior of services ReadDTCInformation for ExtendedData and SnapshotData for suppressed DTCs
[SWS_DM_01305]	Behavior of services ReadDTCInformation with DTC mask record for suppressed DTCs
[SWS_DM_01306]	Functionality of DTCInformation::GetDtcSuppression
[SWS_DM_01307]	Precondition for suppression
[SWS_DM_01308]	Functionality of DTCInformation::SetDtcSuppression
[SWS_DM_01309]	Unblock requests after ara::diag::EcuResetRequest::ExecuteReset completed
[SWS_DM_01310]	Realization of UDS service RequestFileTransfer (0x38)
[SWS_DM_01311]	Realization of modeOfOperation AddFile (0x01)
[SWS_DM_01312]	Realization of modeOfOperation DeleteFile (0x02)
[SWS_DM_01313]	Realization of modeOfOperation ReplaceFile (0x03)
[SWS_DM_01314]	Realization of modeOfOperation ReadFile (0x04)
[SWS_DM_01315]	Realization of modeOfOperation ReadDir (0x05)

▽

△

Number	Heading
[SWS_DM_01316]	Realization of modeOfOperation ResumeFile (0x06)
[SWS_DM_01317]	Realization of TransferData (0x36) in context of RequestFileTransfer
[SWS_DM_01318]	Realization of RequestTransferExit (0x37) in context of RequestFileTransfer
[SWS_DM_01319]	Consecutive registration of notifier with ReleaseHandler::SetNotifier()
[SWS_DM_01320]	
[SWS_DM_01321]	
[SWS_DM_01322]	
[SWS_DM_01323]	
[SWS_DM_01324]	
[SWS_DM_01325]	
[SWS_DM_01326]	
[SWS_DM_01327]	
[SWS_DM_01328]	
[SWS_DM_01329]	
[SWS_DM_01330]	
[SWS_DM_01331]	
[SWS_DM_01332]	
[SWS_DM_01333]	
[SWS_DM_01334]	
[SWS_DM_01335]	
[SWS_DM_01336]	
[SWS_DM_01337]	
[SWS_DM_01339]	
[SWS_DM_01340]	
[SWS_DM_01342]	
[SWS_DM_01343]	
[SWS_DM_01344]	
[SWS_DM_01345]	Lifetime of MetaInfo
[SWS_DM_01346]	Handling negative return values of ara::diag::EcuResetRequest::ExecuteReset
[SWS_DM_01347]	Handling unspecified negative return values of ara::diag::EcuResetRequest::ExecuteReset
[SWS_DM_01348]	Consecutive registration of notifier with CancellationHandler::SetNotifier()
[SWS_DM_01349]	Consecutive registration of notifier with SetEventStatusChangedNotifier()
[SWS_DM_01350]	Consecutive registration of notifier with SetDTCStatusChangedNotifier()
[SWS_DM_01351]	Consecutive registration of notifier with SetSnapshotRecordUpdatedNotifier()
[SWS_DM_01352]	Consecutive registration of notifier with SetNumberOfStoredEntriesNotifier()
[SWS_DM_01353]	Consecutive registration of notifier with SetControlDtcStatusNotifier()

▽

△

Number	Heading
[SWS_DM_01354]	Consecutive registration of notifier with SetEventMemoryOverflowNotifier()
[SWS_DM_01355]	Consecutive registration of notifier with SetActivityNotifier()
[SWS_DM_01356]	Consecutive registration of notifier with SetDiagnosticSessionNotifier()
[SWS_DM_01357]	Consecutive registration of notifier with SetSecurityLevelNotifier()
[SWS_DM_01358]	Consecutive registration of notifier with OperationCycle::SetNotifier()
[SWS_DM_01359]	Consecutive registration of notifier with Indicator::SetNotifier()
[SWS_DM_01360]	Consecutive registration of notifier with ClientAuthentication::SetNotifier()
[SWS_DM_01361]	Providing the EID in DoIP protocol messages
[SWS_DM_01362]	
[SWS_DM_01363]	
[SWS_DM_01364]	
[SWS_DM_01365]	
[SWS_DM_01366]	
[SWS_DM_01367]	
[SWS_DM_01368]	
[SWS_DM_01369]	DM as SOVD Server
[SWS_DM_01370]	DNS-Based Service Discovery and Multicast DNS for SOVD
[SWS_DM_01371]	Secure Communication for SOVD using TLS
[SWS_DM_01372]	Representation of DM by SOVD component
[SWS_DM_01373]	Representation of Diagnostic Server instance by SOVD subcomponents
[SWS_DM_01374]	Dispatching of SOVD requests/responses
[SWS_DM_01375]	Behavior on locked SOVD
[SWS_DM_01376]	Response behavior of SOVD services without access rights
[SWS_DM_01379]	SOVD mode control_dtc_setting set value
[SWS_DM_01380]	SOVD mode control_dtc_setting get value
[SWS_DM_01381]	SOVD mode control_dtc_setting value schema
[SWS_DM_01382]	SOVD mode control_dtc_setting name
[SWS_DM_01383]	SOVD mode control_dtc_setting
[SWS_DM_01384]	SOVD mode communication_control set value
[SWS_DM_01385]	SOVD mode communication_control get value
[SWS_DM_01386]	SOVD mode communication_control value schema
[SWS_DM_01387]	SOVD mode communication_control name
[SWS_DM_01388]	SOVD mode communication_control
[SWS_DM_01389]	SOVD method Retrieve List of All Supported Modes of an Entity
[SWS_DM_01390]	SOVD operations request and response parameters
[SWS_DM_01391]	SOVD method Stop the Execution of an Operation proximity_response

▽

△

Number	Heading
[SWS_DM_01392]	SOVD method `Stop the Execution of an Operation`
[SWS_DM_01393]	SOVD method `Get the Status of an Operation Execution`
[SWS_DM_01394]	SOVD method `Get Executions of an Operation`
[SWS_DM_01395]	SOVD method `Start Execution of an Operation proximity_response`
[SWS_DM_01396]	SOVD method `Start Execution of an Operation`
[SWS_DM_01397]	SOVD operation `mode`
[SWS_DM_01398]	SOVD operation `proximity_challenge`
[SWS_DM_01399]	SOVD operation attribute `asynchronous_execution`
[SWS_DM_01400]	SOVD operation attribute `proximity_proof_required`
[SWS_DM_01401]	SOVD operation attribute `name`
[SWS_DM_01402]	SOVD operation attribute `id`
[SWS_DM_01403]	SOVD method `Get Details of a Single Operation`
[SWS_DM_01404]	SOVD method `Retrieve List of All Available Operations from an Entity`
[SWS_DM_01405]	SOVD operations
[SWS_DM_01406]	SOVD method `Delete Single Fault of an Entity`
[SWS_DM_01407]	SOVD method `Delete All Faults of an Entity` without scope
[SWS_DM_01408]	SOVD method `Delete All Faults of an Entity` with scope
[SWS_DM_01409]	SOVD method `Delete All Faults of an Entity`
[SWS_DM_01411]	SOVD fault `environment_data` query
[SWS_DM_01412]	SOVD fault `environment_data`
[SWS_DM_01413]	SOVD fault attribute `symptom`
[SWS_DM_01414]	SOVD fault attribute `status`
[SWS_DM_01415]	SOVD fault general attributes
[SWS_DM_01416]	SOVD method `Read Details for a Fault`
[SWS_DM_01417]	SOVD method `Read Faults from an Entity`
[SWS_DM_01418]	SOVD faults
[SWS_DM_01419]	SOVD method `Read Multiple Data Values at Once from an Entity Using a Data List`
[SWS_DM_01420]	SOVD data-list
[SWS_DM_01421]	SOVD method `Write Configuration as Parameters` data processing
[SWS_DM_01422]	SOVD method `Write Configuration as Parameters`
[SWS_DM_01423]	SOVD method `Read Configuration as Parameters` data query
[SWS_DM_01424]	SOVD method `Read Configuration as Parameters`
[SWS_DM_01425]	SOVD configuration attribute `version` and `content_type`
[SWS_DM_01426]	SOVD configuration attribute `type`
[SWS_DM_01427]	SOVD configuration attribute `name`
[SWS_DM_01428]	SOVD configuration attribute `id`

▽

△

Number	Heading
[SWS_DM_01429]	SOVD method Retrieve List of All Configurations Provided by the Entity
[SWS_DM_01430]	SOVD method Write a Data Value to an Entity data processing
[SWS_DM_01431]	SOVD method Write a Data Value to an Entity
[SWS_DM_01432]	SOVD method Read Single Data Value from an Entity data query
[SWS_DM_01433]	SOVD method Read Single Data Value from an Entity
[SWS_DM_01434]	SOVD data attribute data of internal structure
[SWS_DM_01435]	SOVD group id
[SWS_DM_01436]	SOVD group attribute category
[SWS_DM_01437]	SOVD group category uniqueness
[SWS_DM_01438]	SOVD data attribute group
[SWS_DM_01439]	SOVD data attribute category
[SWS_DM_01440]	SOVD data attribute name
[SWS_DM_01441]	SOVD data attribute id
[SWS_DM_01442]	SOVD method Retrieve List of All Data Provided by the Entity
[SWS_DM_01443]	SOVD method for locking
[SWS_DM_01444]	Data-groups type content
[SWS_DM_01445]	Data-groups type
[SWS_DM_01446]	SOVD method Retrieve Groups Supported by a Data Resource Collection
[SWS_DM_01447]	Data categories type
[SWS_DM_01448]	Standard resource Data categories
[SWS_DM_01449]	SOVD method Retrieve Categories Supported by a Data Resource Collection
[SWS_DM_01450]	SOVDInfo type content
[SWS_DM_01451]	SOVDInfo type
[SWS_DM_01452]	Mismatching versions
[SWS_DM_01453]	path to version-info
[SWS_DM_01454]	Query to the SOVD API version
[SWS_DM_01455]	SOVD method for SOVD API Versioning
[SWS_DM_01456]	SOVD method Query an Online Capability Description
[SWS_DM_01457]	SOVD data representation of arrays
[SWS_DM_01458]	SOVD data representation of units
[SWS_DM_01459]	SOVD data representation of textual Strings
[SWS_DM_01460]	SOVD data representation of baseTypeEncoding IEEE754
[SWS_DM_01461]	SOVD data representation of BITFIELD_TEXTTABLE
[SWS_DM_01462]	SOVD data representation of TAB_NOINTP
[SWS_DM_01463]	SOVD data representation of atomic scaled numeric data with texttable
[SWS_DM_01464]	SOVD data representation of TEXTTABLE

▽

△

Number	Heading
[SWS_DM_01465]	SOVD data representation of atomic numeric data
[SWS_DM_01466]	Environmental Condition Check for SOVD evaluated to FALSE
[SWS_DM_01467]	Environmental Condition Check for SOVD evaluated to TRUE
[SWS_DM_01468]	Check of Environmental Conditions before executing SOVD methods
[SWS_DM_01469]	Validity period of authenticated roles
[SWS_DM_01470]	Authorization validation
[SWS_DM_01471]	Redirection to token endpoint
[SWS_DM_01472]	Redirection to authorization endpoint
[SWS_DM_01473]	DM shall lock SOVD entity after time expiration
[SWS_DM_01474]	DM shall allow access only to unlocked SOVD entities
[SWS_DM_01475]	DM shall allow only one lock per SOVD entity
[SWS_DM_01476]	Parallel SOVD client handling
[SWS_DM_01477]	SOVD lock in UDS extended session
[SWS_DM_01478]	
[SWS_DM_01479]	
[SWS_DM_01480]	
[SWS_DM_01481]	
[SWS_DM_01482]	
[SWS_DM_01483]	
[SWS_DM_01484]	
[SWS_DM_01485]	
[SWS_DM_01486]	
[SWS_DM_01487]	
[SWS_DM_01488]	
[SWS_DM_01489]	
[SWS_DM_01490]	
[SWS_DM_01491]	
[SWS_DM_01492]	
[SWS_DM_01493]	
[SWS_DM_01494]	
[SWS_DM_01495]	
[SWS_DM_01496]	
[SWS_DM_01497]	
[SWS_DM_01498]	
[SWS_DM_01499]	
[SWS_DM_01500]	
[SWS_DM_01501]	
[SWS_DM_01502]	
[SWS_DM_01503]	

▽

△

Number	Heading
[SWS_DM_01504]	
[SWS_DM_01505]	
[SWS_DM_01506]	
[SWS_DM_01507]	
[SWS_DM_01508]	
[SWS_DM_01509]	
[SWS_DM_01510]	
[SWS_DM_01511]	
[SWS_DM_01512]	
[SWS_DM_01513]	
[SWS_DM_01514]	
[SWS_DM_01515]	
[SWS_DM_01516]	
[SWS_DM_01517]	
[SWS_DM_01518]	
[SWS_DM_01519]	
[SWS_DM_01520]	
[SWS_DM_01521]	
[SWS_DM_01522]	
[SWS_DM_01523]	
[SWS_DM_01524]	
[SWS_DM_01525]	`ara::diag::DoIPPowerMode` not yet offered when client requests DoIP PowerMode
[SWS_DM_01526]	`ara::diag::DoIPActivationLine` not yet offered on activation line controlled network interfaces
[SWS_DM_01527]	`ara::diag::DoIPGroupIdentification` not yet offered when DM needs to retrieve GID
[SWS_DM_01528]	`ara::diag::DoIPEntityIdentification` not yet offered when DM needs to retrieve EID from Application
[SWS_DM_01529]	Behavior on failed ara::diag instantiation
[SWS_DM_01530]	
[SWS_DM_01531]	
[SWS_DM_01532]	
[SWS_DM_01533]	
[SWS_DM_01534]	
[SWS_DM_01535]	
[SWS_DM_01536]	
[SWS_DM_01537]	
[SWS_DM_01538]	
[SWS_DM_01539]	

▽

△

Number	Heading
[SWS_DM_01540]	
[SWS_DM_01541]	
[SWS_DM_01542]	
[SWS_DM_01543]	
[SWS_DM_01544]	
[SWS_DM_01545]	
[SWS_DM_01546]	
[SWS_DM_01547]	
[SWS_DM_01548]	
[SWS_DM_01549]	
[SWS_DM_01550]	
[SWS_DM_01551]	
[SWS_DM_01552]	
[SWS_DM_01553]	
[SWS_DM_01554]	
[SWS_DM_01555]	
[SWS_DM_01556]	
[SWS_DM_01557]	
[SWS_DM_01558]	
[SWS_DM_01559]	
[SWS_DM_01560]	
[SWS_DM_01561]	Response Body forSOVD fault environment_data table

Table E.22: Added Specification Items in R22-11

E.8.2 Changed Specification Items in R22-11

Number	Heading
[SWS_DM_00005]	DoIP Support
[SWS_DM_00055]	Supported event memories
[SWS_DM_00062]	Mapping between ISO 14229-1 and Autosar Diagnostic Extract Template of the DTCFormatIdentifier
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00126]	Realisation of UDS service 0x3E TesterPresent
[SWS_DM_00128]	Realization of UDS service RequestDownload (0x34)

▽

△

Number	Heading
[SWS_DM_00134]	Realization of UDS service RequestUpload (0x35)
[SWS_DM_00137]	Realization of UDS service TransferData (0x36)
[SWS_DM_00140]	Realisation of UDS service 0x28 CommunicationControl
[SWS_DM_00141]	Realization of UDS service RequestTransferExit (0x37)
[SWS_DM_00159]	Allow only to clear GroupOfAllDTCs
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00170]	Realisation of UDS service ReadDataByIdentifier (0x22)
[SWS_DM_00186]	Realisation of UDS service WriteDataByIdentifier (0x2E)
[SWS_DM_00201]	Realization of UDS service RoutineControl (0x31)
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00213]	DTC status processing
[SWS_DM_00217]	UDS DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	kConfirmedDTC (Bit3) calculation
[SWS_DM_00226]	Support of UDS service DiagnosticSessionControl
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00229]	Support of UDS service ControlDTCSetting (0x85)
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00252]	Reaction on Unsupported Subfunction
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00277]	Cancellation of a Diagnostic Conversation in case of External Service Processing
[SWS_DM_00278]	Cancellation of a Diagnostic Conversation in case of Internal Processing
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	

▽

△

Number	Heading
[SWS_DM_00298]	
[SWS_DM_00299]	
[SWS_DM_00300]	
[SWS_DM_00301]	
[SWS_DM_00302]	
[SWS_DM_00303]	
[SWS_DM_00304]	
[SWS_DM_00306]	
[SWS_DM_00307]	
[SWS_DM_00309]	
[SWS_DM_00310]	
[SWS_DM_00311]	
[SWS_DM_00312]	
[SWS_DM_00313]	
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	
[SWS_DM_00326]	
[SWS_DM_00327]	
[SWS_DM_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00363]	Unsupported Subfunction
[SWS_DM_00368]	DM takes care of Response Pending Messages
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00384]	
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00431]	Replacement of Diagnostic Conversations

▽

△

Number	Heading
[SWS_DM_00448]	Check supported RoutineIdentifier in active session
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00475]	DoIP Version
[SWS_DM_00480]	Security Access Blocking Timer
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	
[SWS_DM_00521]	
[SWS_DM_00522]	
[SWS_DM_00523]	
[SWS_DM_00524]	
[SWS_DM_00525]	
[SWS_DM_00526]	
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	

▽

△

Number	Heading
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing with generic interface
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing with generic interface
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing with generic interface
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	

▽

△

Number	Heading
[SWS_DM_00599]	
[SWS_DM_00600]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00617]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00620]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	

▽

△

Number	Heading
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00644]	
[SWS_DM_00645]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	
[SWS_DM_00658]	
[SWS_DM_00659]	
[SWS_DM_00660]	
[SWS_DM_00661]	
[SWS_DM_00662]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00690]	
[SWS_DM_00691]	
[SWS_DM_00692]	

▽

△

Number	Heading
[SWS_DM_00693]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	
[SWS_DM_00710]	
[SWS_DM_00711]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00730]	
[SWS_DM_00731]	
[SWS_DM_00732]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	

▽

△

Number	Heading
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00755]	
[SWS_DM_00760]	
[SWS_DM_00761]	
[SWS_DM_00762]	
[SWS_DM_00763]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00767]	
[SWS_DM_00770]	
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00784]	
[SWS_DM_00785]	
[SWS_DM_00786]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00793]	
[SWS_DM_00794]	
[SWS_DM_00795]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	

▽

△

Number	Heading
[SWS_DM_00802]	
[SWS_DM_00803]	
[SWS_DM_00804]	
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	
[SWS_DM_00809]	
[SWS_DM_00810]	
[SWS_DM_00820]	
[SWS_DM_00821]	
[SWS_DM_00822]	
[SWS_DM_00830]	
[SWS_DM_00831]	
[SWS_DM_00832]	
[SWS_DM_00833]	
[SWS_DM_00834]	
[SWS_DM_00835]	
[SWS_DM_00836]	
[SWS_DM_00837]	
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00860]	No service processing
[SWS_DM_00863]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00864]	Checking Supported Subfunction for CompareKey
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00933]	UDS DTC status bit 5 / 'testFailedSinceLastClear' after displacement
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00945]	Occurrence Counter initial value
[SWS_DM_00946]	Occurrence Counter increment strategy 'testFailed'-only
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00967]	Support of UDS service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00971]	

▽

△

Number	Heading
[SWS_DM_00972]	
[SWS_DM_00973]	
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00984]	Return of cancellation status
[SWS_DM_00989]	
[SWS_DM_00990]	
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after `ara::diag::EcuResetRequest::RequestReset` called

▽

△

Number	Heading
[SWS_DM_01024]	Event Status processing
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_01075]	Maximum number of sourceDataIdentifiers in the request
[SWS_DM_01088]	
[SWS_DM_01089]	
[SWS_DM_01092]	EnableRapidPowerShutdown Positive Response
[SWS_DM_01094]	Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On
[SWS_DM_01095]	Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off
[SWS_DM_01100]	Debouncing Algorithm not fitting
[SWS_DM_01102]	
[SWS_DM_01118]	Support of DEXT parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate
[SWS_DM_01119]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentfierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength
[SWS_DM_01122]	Support of DEXT parameter DiagnosticResponseOnEvent-Class.maxNumberOfStoredDTCStatusChangedEvents
[SWS_DM_01123]	
[SWS_DM_01124]	
[SWS_DM_01125]	
[SWS_DM_01126]	
[SWS_DM_01127]	
[SWS_DM_01128]	
[SWS_DM_01129]	
[SWS_DM_01130]	
[SWS_DM_01131]	
[SWS_DM_01132]	
[SWS_DM_01133]	
[SWS_DM_01134]	
[SWS_DM_01136]	

▽

△

Number	Heading
[SWS_DM_01137]	
[SWS_DM_01138]	
[SWS_DM_01139]	
[SWS_DM_01140]	
[SWS_DM_01141]	
[SWS_DM_01142]	
[SWS_DM_01143]	
[SWS_DM_01144]	
[SWS_DM_01145]	
[SWS_DM_01146]	
[SWS_DM_01147]	
[SWS_DM_01148]	
[SWS_DM_01149]	
[SWS_DM_01150]	
[SWS_DM_01151]	
[SWS_DM_01152]	
[SWS_DM_01153]	
[SWS_DM_01154]	
[SWS_DM_01155]	
[SWS_DM_01156]	
[SWS_DM_01157]	
[SWS_DM_01158]	
[SWS_DM_01159]	
[SWS_DM_01160]	
[SWS_DM_01161]	
[SWS_DM_01162]	
[SWS_DM_01163]	
[SWS_DM_01164]	
[SWS_DM_01165]	
[SWS_DM_01166]	
[SWS_DM_01167]	
[SWS_DM_01168]	
[SWS_DM_01169]	
[SWS_DM_01170]	
[SWS_DM_01171]	
[SWS_DM_01172]	
[SWS_DM_01173]	
[SWS_DM_01174]	
[SWS_DM_01175]	

▽

△

Number	Heading
[SWS_DM_01176]	
[SWS_DM_01177]	
[SWS_DM_01178]	
[SWS_DM_01179]	
[SWS_DM_01180]	
[SWS_DM_01181]	
[SWS_DM_01182]	
[SWS_DM_01183]	
[SWS_DM_01184]	
[SWS_DM_01185]	
[SWS_DM_01186]	
[SWS_DM_01187]	
[SWS_DM_01188]	
[SWS_DM_01189]	
[SWS_DM_01190]	
[SWS_DM_01191]	
[SWS_DM_01192]	
[SWS_DM_01193]	
[SWS_DM_01194]	
[SWS_DM_01195]	
[SWS_DM_01196]	
[SWS_DM_01197]	
[SWS_DM_01198]	
[SWS_DM_01199]	
[SWS_DM_01200]	
[SWS_DM_01201]	
[SWS_DM_01247]	Validation of the transmitCertificate certificateEvaluationId
[SWS_DM_09010]	
[SWS_DM_09012]	
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_09017]	
[SWS_DM_09021]	
[SWS_DM_NA]	

Table E.23: Changed Specification Items in R22-11

E.8.3 Deleted Specification Items in R22-11

Number	Heading
[SWS_DM_00007]	Uniqueness of diagnostic events
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00262]	Common semantic behavior for ClearDTC triggered via diagnostics or application
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00365]	Suppression of positive response in accordance to ISO 14229-1[17]
[SWS_DM_00376]	Positive response processing
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00857]	Signature of Manufacturer Permission Check Method
[SWS_DM_00858]	Signature of Supplier Permission Check Method
[SWS_DM_00861]	Negative response processing
[SWS_DM_00862]	Suppression of negative response for functional requests in accordance to ISO 14229-1[17]
[SWS_DM_00873]	Diagnostic event processing interface
[SWS_DM_01084]	Triggering for snapshot record storage (calculated)
[SWS_DM_01091]	Maximum time frame to trigger ExecuteReset()
[SWS_DM_09011]	
[SWS_DM_09013]	
[SWS_DM_09014]	
[SWS_DM_09018]	

Table E.24: Deleted Specification Items in R22-11

E.8.4 Added Constraints in R22-11

none

E.8.5 Changed Constraints in R22-11

none

E.8.6 Deleted Constraints in R22-11

none

E.9 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11

E.9.1 Added Specification Items in R23-11

Number	Heading
[SWS_DM_00796]	Definition of API variable ara::diag::UploadService::Operation Output::responseData
[SWS_DM_00985]	Definition of API class ara::diag::DiagOfferException
[SWS_DM_00986]	Definition of API function ara::diag::DiagOfferException::DiagOfferException
[SWS_DM_00987]	Definition of API class ara::diag::DiagReportingException
[SWS_DM_00988]	Definition of API function ara::diag::DiagReportingException::DiagReporting Exception
[SWS_DM_01003]	Definition of API function ara::diag::GetDiagOfferDomain
[SWS_DM_01004]	Definition of API function ara::diag::GetDiagReportingDomain
[SWS_DM_01562]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::controlType
[SWS_DM_01563]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::communicationType
[SWS_DM_01564]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::nodeIdentificationNumber
[SWS_DM_01565]	internaldataelements
[SWS_DM_01566]	
[SWS_DM_01567]	SovdFaultAttributes
[SWS_DM_01568]	supportedDoIpMessageTypes
[SWS_DM_01569]	Configurable reset of the pendingDTC bit in case of displacement
[SWS_DM_01570]	Lifetime of overridden default roles
[SWS_DM_01571]	Load persisted data
[SWS_DM_01572]	Graceful shutdown
[SWS_DM_01573]	Stop all running `Transport Protocol Handler`s
[SWS_DM_01574]	Write data to be persisted
[SWS_DM_01575]	Configuration of permanent storage of "TestFailed" status bits
[SWS_DM_01576]	Behavior of not configured `DiagnosticMemoryDestination.agingRequiresTestedCycle`
[SWS_DM_01577]	Behavior of not configured `DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear`
[SWS_DM_01578]	Behavior of not configured `DiagnosticMemoryDestination.clearDtcLimitation`
[SWS_DM_01579]	Behavior of not configured `DiagnosticMemoryDestination.memoryEntryStorageTrigger`
[SWS_DM_01580]	

▽

△

Number	Heading
[SWS_DM_01581]	Assigning a UDS request to a new Diagnostic Conversation in active default session
[SWS_DM_01582]	Healing counter increment
[SWS_DM_01583]	Resetting counter-based debouncing
[SWS_DM_01584]	Resetting time-based debouncing
[SWS_DM_01596]	Definition of API function ara::diag::OperationCycle::operator=
[SWS_DM_01597]	Definition of API function ara::diag::OperationCycle::operator=
[SWS_DM_01598]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_01599]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_01607]	Definition of API function ara::diag::Authentication::operator=
[SWS_DM_01608]	Definition of API function ara::diag::Authentication::operator=
[SWS_DM_01609]	Definition of API function ara::diag::Authentication::Authentication
[SWS_DM_01610]	Definition of API function ara::diag::Authentication::Authentication
[SWS_DM_01611]	Definition of API function ara::diag::DTCInformation::operator=
[SWS_DM_01612]	Definition of API function ara::diag::DTCInformation::operator=
[SWS_DM_01613]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_01614]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_01615]	Definition of API function ara::diag::SecurityAccess::operator=
[SWS_DM_01616]	Definition of API function ara::diag::SecurityAccess::operator=
[SWS_DM_01617]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_01618]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_01619]	Definition of API function ara::diag::EcuResetRequest::operator=
[SWS_DM_01620]	Definition of API function ara::diag::EcuResetRequest::operator=
[SWS_DM_01621]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01622]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01623]	Definition of API function ara::diag::Condition::operator=
[SWS_DM_01624]	Definition of API function ara::diag::Condition::operator=
[SWS_DM_01625]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_01626]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_01627]	Definition of API function ara::diag::Indicator::operator=
[SWS_DM_01628]	Definition of API function ara::diag::Indicator::operator=
[SWS_DM_01629]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_01630]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_01631]	Definition of API function ara::diag::DoIPActivationLine::operator=
[SWS_DM_01632]	Definition of API function ara::diag::DoIPActivationLine::operator=
[SWS_DM_01633]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_01634]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_01635]	Definition of API function ara::diag::GenericRoutine::operator=
[SWS_DM_01636]	Definition of API function ara::diag::GenericRoutine::operator=

▽

△

Number	Heading
[SWS_DM_01637]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_01638]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_01639]	Definition of API function ara::diag::DoIPGroupIdentification::operator=
[SWS_DM_01640]	Definition of API function ara::diag::DoIPGroupIdentification::operator=
[SWS_DM_01641]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroup Identification
[SWS_DM_01642]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroup Identification
[SWS_DM_01643]	Definition of API function ara::diag::DoIPPowerMode::operator=
[SWS_DM_01644]	Definition of API function ara::diag::DoIPPowerMode::operator=
[SWS_DM_01645]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_01646]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_01647]	Definition of API function ara::diag::DownloadService::operator=
[SWS_DM_01648]	Definition of API function ara::diag::DownloadService::operator=
[SWS_DM_01649]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01650]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01651]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01652]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01653]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01654]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01655]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01656]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01657]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01658]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01659]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::operator=
[SWS_DM_01660]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::operator=
[SWS_DM_01661]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface
[SWS_DM_01662]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface
[SWS_DM_01663]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::operator=
[SWS_DM_01664]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::operator=
[SWS_DM_01665]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface

▽

△

Number	Heading
[SWS_DM_01666]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface
[SWS_DM_01667]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::operator=
[SWS_DM_01668]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::operator=
[SWS_DM_01669]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::ShortnameOfRIPortInterface
[SWS_DM_01670]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::ShortnameOfRIPortInterface
[SWS_DM_01671]	Definition of API function ara::diag::UploadService::operator=
[SWS_DM_01672]	Definition of API function ara::diag::UploadService::operator=
[SWS_DM_01673]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_01674]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_01675]	Definition of API function ara::diag::CommunicationControl::operator=
[SWS_DM_01676]	Definition of API function ara::diag::CommunicationControl::operator=
[SWS_DM_01677]	Definition of API function ara::diag::CommunicationControl::Communication Control
[SWS_DM_01678]	Definition of API function ara::diag::CommunicationControl::Communication Control
[SWS_DM_01679]	Definition of API function ara::diag::Event::operator=
[SWS_DM_01680]	Definition of API function ara::diag::Event::operator=
[SWS_DM_01681]	Definition of API function ara::diag::Event::Event
[SWS_DM_01682]	Definition of API function ara::diag::Event::Event
[SWS_DM_01683]	Definition of API function ara::diag::Monitor::operator=
[SWS_DM_01684]	Definition of API function ara::diag::Monitor::operator=
[SWS_DM_01685]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01686]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01687]	Definition of API function ara::diag::ServiceValidation::operator=
[SWS_DM_01688]	Definition of API function ara::diag::ServiceValidation::operator=
[SWS_DM_01689]	Definition of API function ara::diag::ServiceValidation::ServiceValidation
[SWS_DM_01690]	Definition of API function ara::diag::ServiceValidation::ServiceValidation
[SWS_DM_01694]	Definition of API class ara::diag::MultipleMonitor
[SWS_DM_01695]	Definition of API function ara::diag::MultipleMonitor::MultipleMonitor
[SWS_DM_01696]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01697]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01698]	Definition of API function ara::diag::MultipleMonitor::ReportMonitorAction
[SWS_DM_01699]	Definition of API function ara::diag::MultipleMonitor::Offer
[SWS_DM_01700]	Definition of API function ara::diag::MultipleMonitor::StopOffer
[SWS_DM_01701]	Definition of API type ara::diag::MonitorHandleType

▽

△

Number	Heading
[SWS_DM_01702]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01703]	Definition of API type ara::diag::EventHandleType
[SWS_DM_01704]	Definition of API class ara::diag::MultipleEvent
[SWS_DM_01705]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_01706]	Definition of API function ara::diag::MultipleEvent::~MultipleEvent
[SWS_DM_01707]	Definition of API function ara::diag::MultipleEvent::GetEventStatus
[SWS_DM_01708]	Definition of API function ara::diag::MultipleEvent::SetEventStatusChanged Notifier
[SWS_DM_01709]	Definition of API function ara::diag::MultipleEvent::GetDTCNumber
[SWS_DM_01710]	Definition of API function ara::diag::MultipleEvent::GetDebouncingStatus
[SWS_DM_01711]	Definition of API function ara::diag::MultipleEvent::GetFaultDetectionCounter
[SWS_DM_01726]	Definition of API type ara::diag::ConditionHandleType
[SWS_DM_01727]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_01728]	Definition of API function ara::diag::MultipleCondition::GetCondition
[SWS_DM_01729]	Definition of API function ara::diag::MultipleCondition::SetCondition
[SWS_DM_01730]	Definition of API class ara::diag::MultipleCondition
[SWS_DM_01731]	MultipleMonitor MonitorHandleType Generation
[SWS_DM_01732]	Invalid MonitorHandleType
[SWS_DM_01733]	MultipleEvent EventHandleType Generation
[SWS_DM_01734]	Invalid EventHandleType
[SWS_DM_01735]	MultipleCondition ConditionHandleType Generation
[SWS_DM_01736]	Invalid ConditionHandleType
[SWS_DM_01737]	Definition of API function ara::diag::MultipleCondition::~MultipleCondition
[SWS_DM_01739]	Authentication disabled
[SWS_DM_01740]	Initial enable condition state with notifier callback for disabled enable conditions
[SWS_DM_01741]	No more method calls after stop
[SWS_DM_01742]	Asynchronous UDS protocol start
[SWS_DM_01743]	Require a started UdsTransportProtocolHandler
[SWS_DM_01744]	Processing of ChannelReestablished
[SWS_DM_01745]	Behavior on failed transport protocol initialization
[SWS_DM_01746]	Required successful initialization
[SWS_DM_01747]	S3 timer value
[SWS_DM_01749]	Definition of API function ara::diag::EventStatusByte::operator=
[SWS_DM_01750]	Definition of API function ara::diag::EventStatusByte::operator=
[SWS_DM_01751]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01752]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01753]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01754]	Definition of API function ara::diag::EventStatusByte::IsNotSet

▽

△

Number	Heading
[SWS_DM_01755]	Definition of API function ara::diag::EventStatusByte::IsSet
[SWS_DM_01756]	Definition of API function ara::diag::EventStatusByte::IsPassedAndTested
[SWS_DM_01757]	Definition of API function ara::diag::EventStatusByte::IsFailedAndTested
[SWS_DM_01758]	Security Access Delay Timer on power up
[SWS_DM_01759]	DiagnosticDataElement serialization respecting the data element endianness for typed interfaces
[SWS_DM_01760]	Security events for Diagnostic Management
[SWS_DM_01761]	Reporting 'Request out of range' to IdsM
[SWS_DM_01762]	Reporting 'Sequence error' to IdsM
[SWS_DM_01763]	Reporting 'Data written using WriteDataByIdentifer' to IdsM
[SWS_DM_01764]	Reporting 'Writing invalid data is requested' to IdsM
[SWS_DM_01765]	Reporting 'Download sequence requested' to IdsM
[SWS_DM_01766]	Reporting 'ECU reset triggered ECUReset' to IdsM
[SWS_DM_01767]	Reporting 'Communication control switched off' to IdsM
[SWS_DM_01768]	Reporting 'DTC fault memory cleared' to IdsM
[SWS_DM_01769]	Reporting 'DTC setting switched off' to IdsM
[SWS_DM_01770]	Reporting 'successful authentication' to IdsM
[SWS_DM_01771]	Reporting 'certificate failed' to IdsM
[SWS_DM_01772]	Reporting 'authentication required' to IdsM
[SWS_DM_01773]	Reporting 'required time delay not expired for security access' to IdsM
[SWS_DM_01774]	Reporting 'number of attempts exceeded for security access' to IdsM
[SWS_DM_01775]	Reporting 'security access with invalid key' to IdsM
[SWS_DM_01776]	Reporting 'security unlocked successfully for security access' to IdsM
[SWS_DM_01777]	Reporting 'security access denied' to IdsM
[SWS_DM_01778]	Reporting 'incorrect message length or invalid format' to IdsM
[SWS_DM_01779]	Reporting 'service subfunction not supported' to IdsM
[SWS_DM_01780]	Reporting service not supported to IdsM
[SWS_DM_01781]	(Re-)Identification of Clients by Token
[SWS_DM_01782]	(Re-)Identification of Clients by Identity
[SWS_DM_01783]	Locking only for authorized SOVD clients
[SWS_DM_01784]	Service Validation of SOVD Requests
[SWS_DM_01785]	Service Validation for SOVD Resource Collections
[SWS_DM_01786]	SOVD `Online Capability Description` role sensitivity
[SWS_DM_01787]	Realization of SOVD `Read Configuration`
[SWS_DM_01788]	Realization of SOVD `Write Configuration`
[SWS_DM_01789]	Realization of SOVD `API Methods for Handling of Bulk Data`
[SWS_DM_01790]	Realization of SOVD `Bulk Data API Method for Retrieve List of all Bulk Data Categories`
[SWS_DM_01791]	Realization of SOVD `Read Bulk Data Meta Data`

▽

△

Number	Heading
[SWS_DM_01792]	Realization of SOVD Download Bulk Data
[SWS_DM_01793]	Realization of SOVD Upload Bulk Data
[SWS_DM_01794]	Realization of SOVD Delete All Bulk Data Defined by Category
[SWS_DM_01795]	Realization of SOVD Delete Specific Bulk Data Resource
[SWS_DM_01796]	Realization of SOVD API Methods for Software Update
[SWS_DM_01797]	Realization of SOVD Retrieve List of All Updates
[SWS_DM_01798]	Realization of SOVD Get Details of Update
[SWS_DM_01799]	Realization of SOVD Automated Installation of an Update
[SWS_DM_01800]	Realization of SOVD Prepare Installation of an Update
[SWS_DM_01801]	Realization of SOVD Execute Installation of an Update
[SWS_DM_01802]	Realization of SOVD Get Status of an Update
[SWS_DM_01803]	Realization of SOVD Delete Update Package from an SOVD Server
[SWS_DM_01804]	Realization of SOVD Register an Update at the SOVD Server
[SWS_DM_01805]	Definition SOVD ara::diag errors
[SWS_DM_01806]	Definition of API enum ara::diag::DiagSovdErrc
[SWS_DM_01807]	Definition of API class ara::diag::DiagSovdException
[SWS_DM_01808]	Definition of API function ara::diag::DiagSovdException::DiagSovdException
[SWS_DM_01809]	Definition of API class ara::diag::DiagSovdErrorDomain
[SWS_DM_01810]	Definition of API type ara::diag::DiagSovdErrorDomain::Errc
[SWS_DM_01811]	Definition of API type ara::diag::DiagSovdErrorDomain::Exception
[SWS_DM_01812]	Definition of API function ara::diag::DiagSovdErrorDomain::DiagSovdError Domain
[SWS_DM_01813]	Definition of API function ara::diag::DiagSovdErrorDomain::Name
[SWS_DM_01814]	Definition of API function ara::diag::DiagSovdErrorDomain::Message
[SWS_DM_01815]	Definition of API function ara::diag::DiagSovdErrorDomain::ThrowAs Exception
[SWS_DM_01816]	Definition of API function ara::diag::GetDiagSovdErrorDomain
[SWS_DM_01817]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_01818]	Definition of API variable ara::diag::MetaInfo::kSovd
[SWS_DM_01819]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorization Output::identity
[SWS_DM_01820]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorization Output::validUntil
[SWS_DM_01821]	Definition of API enum ara::diag::SovdRequestMethod
[SWS_DM_01822]	Definition of API class ara::diag::SovdServiceValidation
[SWS_DM_01823]	Definition of API function ara::diag::SovdServiceValidation::SovdService Validation
[SWS_DM_01824]	Definition of API function ara::diag::SovdServiceValidation::~SovdService Validation
[SWS_DM_01825]	Definition of API function ara::diag::SovdServiceValidation::Offer

▽

△

Number	Heading
[SWS_DM_01826]	Definition of API function ara::diag::SovdServiceValidation::StopOffer
[SWS_DM_01827]	Definition of API function ara::diag::SovdServiceValidation::Validate
[SWS_DM_01828]	Definition of API class ara::diag::HttpRedirect
[SWS_DM_01829]	Definition of API variable ara::diag::HttpRedirect::url
[SWS_DM_01830]	Definition of API class ara::diag::SovdBulkData::BulkDataMetaData Descriptor
[SWS_DM_01831]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::id
[SWS_DM_01832]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::mimetype
[SWS_DM_01833]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::name
[SWS_DM_01834]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::translation_id
[SWS_DM_01835]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::size
[SWS_DM_01836]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::file_name
[SWS_DM_01837]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::creation_date
[SWS_DM_01838]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::last_modified
[SWS_DM_01839]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash
[SWS_DM_01840]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash_algorithm
[SWS_DM_01841]	Definition of API class ara::diag::SovdBulkData
[SWS_DM_01842]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::deleted_items
[SWS_DM_01843]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::failed_items
[SWS_DM_01844]	Definition of API class ara::diag::SovdBulkData::DeletionError
[SWS_DM_01845]	Definition of API variable ara::diag::SovdBulkData::DeletionError::item
[SWS_DM_01846]	Definition of API variable ara::diag::SovdBulkData::DeletionError::error
[SWS_DM_01847]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01848]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01849]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01850]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01851]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01852]	Definition of API function ara::diag::SovdBulkData::~SovdBulkData
[SWS_DM_01853]	Definition of API function ara::diag::SovdBulkData::GetBulkDataMetaData

▽

△

Number	Heading
[SWS_DM_01854]	Definition of API function ara::diag::SovdBulkData::RequestBulkDataUpload
[SWS_DM_01855]	Definition of API function ara::diag::SovdBulkData::DeleteAllBulkData
[SWS_DM_01856]	Definition of API function ara::diag::SovdBulkData::DeleteSpecificBulkData
[SWS_DM_01857]	Definition of API function ara::diag::SovdBulkData::Offer
[SWS_DM_01858]	Definition of API function ara::diag::SovdBulkData::StopOffer
[SWS_DM_01860]	Definition of API class ara::diag::SovdConfiguration
[SWS_DM_01861]	Definition of API class ara::diag::SovdConfiguration::SovdConfigurationMeta Info
[SWS_DM_01862]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::size
[SWS_DM_01863]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::mimetype
[SWS_DM_01864]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01865]	Definition of API function ara::diag::SovdConfiguration::~SovdConfiguration
[SWS_DM_01866]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01867]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01868]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01869]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01870]	Definition of API function ara::diag::SovdConfiguration::RequestGet Configuration
[SWS_DM_01871]	Definition of API function ara::diag::SovdConfiguration::RequestPut Configuration
[SWS_DM_01872]	Definition of API function ara::diag::SovdConfiguration::Offer
[SWS_DM_01873]	Definition of API function ara::diag::SovdBulkData::RequestBulkData Download
[SWS_DM_01874]	Definition of API class ara::diag::SovdSwUpdate
[SWS_DM_01875]	Definition of API enum ara::diag::SovdUpdatePhase
[SWS_DM_01876]	Definition of API enum ara::diag::SovdUpdateStatus
[SWS_DM_01877]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageDetails
[SWS_DM_01878]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::id
[SWS_DM_01879]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::name
[SWS_DM_01881]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::translation_id
[SWS_DM_01882]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::automated
[SWS_DM_01883]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::origin
[SWS_DM_01884]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes
[SWS_DM_01885]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes_translation_id

▽

△

Number	Heading
[SWS_DM_01886]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity
[SWS_DM_01887]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity_translation_id
[SWS_DM_01888]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions
[SWS_DM_01889]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions_translation_id
[SWS_DM_01890]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::execution_conditions
[SWS_DM_01891]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::duration
[SWS_DM_01893]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::size
[SWS_DM_01894]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::updated_components
[SWS_DM_01895]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::affected_components
[SWS_DM_01896]	Definition of API class ara::diag::SovdSwUpdate::SubProgress
[SWS_DM_01898]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::entity
[SWS_DM_01899]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::status
[SWS_DM_01900]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::progress
[SWS_DM_01901]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::error
[SWS_DM_01902]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageStatus
[SWS_DM_01903]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::phase
[SWS_DM_01904]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::status
[SWS_DM_01905]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::progress
[SWS_DM_01906]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::subprogress
[SWS_DM_01907]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step
[SWS_DM_01908]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step_translation_id
[SWS_DM_01909]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::error
[SWS_DM_01910]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01911]	Definition of API function ara::diag::SovdSwUpdate::~SovdSwUpdate
[SWS_DM_01912]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01913]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01914]	Definition of API function ara::diag::SovdSwUpdate::operator=

▽

△

Number	Heading
[SWS_DM_01915]	Definition of API function ara::diag::SovdSwUpdate::operator=
[SWS_DM_01916]	Definition of API function ara::diag::SovdSwUpdate::GetAllUpdates
[SWS_DM_01917]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Details
[SWS_DM_01918]	Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackage Automated
[SWS_DM_01919]	Definition of API function ara::diag::SovdSwUpdate::PrepareUpdatePackage
[SWS_DM_01920]	Definition of API function ara::diag::SovdSwUpdate::ExecuteUpdatePackage
[SWS_DM_01921]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Status
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackage Registration
[SWS_DM_01924]	Definition of API function ara::diag::SovdSwUpdate::Offer
[SWS_DM_01925]	Definition of API function ara::diag::SovdSwUpdate::StopOffer
[SWS_DM_01926]	Definition of API class ara::diag::SovdBulkData::DeleteByCategoryResult
[SWS_DM_01927]	Definition of API class ara::diag::SovdAuthorization::ValidateAuthorization Output
[SWS_DM_01928]	SOVD method for logging
[SWS_DM_01929]	SOVD Error for non-reentrant software
[SWS_DM_01930]	Definition of API function ara::diag::SovdConfiguration::StopOffer
[SWS_DM_01931]	
[SWS_DM_01932]	
[SWS_DM_01933]	
[SWS_DM_01934]	
[SWS_DM_01935]	
[SWS_DM_01936]	
[SWS_DM_01937]	
[SWS_DM_01938]	
[SWS_DM_01939]	
[SWS_DM_01940]	
[SWS_DM_01941]	
[SWS_DM_01942]	
[SWS_DM_01943]	
[SWS_DM_01944]	
[SWS_DM_01945]	
[SWS_DM_01946]	
[SWS_DM_01947]	
[SWS_DM_01948]	
[SWS_DM_01949]	

▽

△

Number	Heading
[SWS_DM_09025]	Definition of API variable ara::diag::uds_transport::UdsTransportProtocol Handler::transportprotocolManager

Table E.25: Added Specification Items in R23-11

E.9.2 Changed Specification Items in R23-11

Number	Heading
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00033]	Debounce timer behavior upon reported `kFailed`
[SWS_DM_00036]	Debounce timer behavior upon reported `kPassed`
[SWS_DM_00085]	Internal debounce timer init
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing
[SWS_DM_00237]	Aging
[SWS_DM_00243]	`Aging`-related `UDS DTC status byte` processing
[SWS_DM_00300]	Definition of API function ara::diag::uds_transport::UdsMessage::GetPayload
[SWS_DM_00326]	Definition of API function ara::diag::uds_transport::UdsTransportProtocol Handler::NotifyReestablishment
[SWS_DM_00331]	Initialization of an `UDS Transport Protocol` implementation
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00390]	Dispatching physical Request
[SWS_DM_00391]	Dispatching functional Request
[SWS_DM_00392]	Properties of returned UdsMessage
[SWS_DM_00393]	Retrieving data for `internal DiagnosticDataElements`
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00479]	Security Access Delay Timer on power up when attempt counter threshold is reached
[SWS_DM_00480]	Shared Security Access Delay Timer
[SWS_DM_00538]	Definition of API class ara::diag::CounterBased
[SWS_DM_00539]	Definition of API class ara::diag::TimeBased
[SWS_DM_00540]	Definition of API enum ara::diag::InitMonitorReason
[SWS_DM_00541]	Definition of API enum ara::diag::MonitorAction
[SWS_DM_00554]	Definition of API function ara::diag::GenericRoutine::Start
[SWS_DM_00555]	Definition of API function ara::diag::GenericRoutine::Stop

▽

△

Number	Heading
[SWS_DM_00556]	Definition of API function ara::diag::GenericRoutine::RequestResults
[SWS_DM_00557]	Definition of API function ara::diag::GenericRoutine::Offer
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00591]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::Start
[SWS_DM_00592]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::Stop
[SWS_DM_00593]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::RequestResults
[SWS_DM_00594]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfRIPortInterface::Offer
[SWS_DM_00596]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::Read
[SWS_DM_00597]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::Offer
[SWS_DM_00598]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::Write
[SWS_DM_00599]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::Offer
[SWS_DM_00610]	Definition of API function ara::diag::CancellationHandler::Cancellation Handler
[SWS_DM_00611]	Definition of API function ara::diag::CancellationHandler::Cancellation Handler
[SWS_DM_00612]	Definition of API function ara::diag::CancellationHandler::operator=
[SWS_DM_00613]	Definition of API function ara::diag::CancellationHandler::operator=
[SWS_DM_00618]	Definition of API function ara::diag::GenericUDSService::HandleMessage
[SWS_DM_00619]	Definition of API function ara::diag::GenericUDSService::Offer
[SWS_DM_00621]	Definition of API variable ara::diag::CounterBased::failedThreshold
[SWS_DM_00622]	Definition of API variable ara::diag::CounterBased::passedThreshold
[SWS_DM_00623]	Definition of API variable ara::diag::CounterBased::failedStepsize
[SWS_DM_00624]	Definition of API variable ara::diag::CounterBased::passedStepsize
[SWS_DM_00625]	Definition of API variable ara::diag::CounterBased::failedJumpValue
[SWS_DM_00626]	Definition of API variable ara::diag::CounterBased::passedJumpValue
[SWS_DM_00627]	Definition of API variable ara::diag::CounterBased::useJumpToFailed
[SWS_DM_00628]	Definition of API variable ara::diag::CounterBased::useJumpToPassed
[SWS_DM_00629]	Definition of API variable ara::diag::TimeBased::failedMs
[SWS_DM_00630]	Definition of API variable ara::diag::TimeBased::passedMs
[SWS_DM_00636]	Definition of API function ara::diag::GenericDataIdentifier::Read
[SWS_DM_00637]	Definition of API function ara::diag::GenericDataIdentifier::Write
[SWS_DM_00638]	Definition of API function ara::diag::GenericDataIdentifier::Offer

▽

△

Number	Heading
[SWS_DM_00640]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::Read
[SWS_DM_00642]	Definition of API enum ara::diag::DTCFormatType
[SWS_DM_00643]	Definition of API enum ara::diag::EventStatusBit
[SWS_DM_00644]	Definition of API class ara::diag::EventStatusByte
[SWS_DM_00645]	Definition of API enum ara::diag::DebouncingState
[SWS_DM_00667]	Definition of API function ara::diag::DTCInformation::SetDTCStatusChanged Notifier
[SWS_DM_00674]	Definition of API function ara::diag::DTCInformation::EnableControlDtc
[SWS_DM_00725]	Definition of API function ara::diag::DoIPGroupIdentification::Offer
[SWS_DM_00735]	Definition of API function ara::diag::DoIPPowerMode::Offer
[SWS_DM_00764]	Definition of API function ara::diag::SecurityAccess::GetSeed
[SWS_DM_00765]	Definition of API function ara::diag::SecurityAccess::CompareKey
[SWS_DM_00766]	Definition of API function ara::diag::SecurityAccess::Offer
[SWS_DM_00774]	Definition of API function ara::diag::ServiceValidation::Validate
[SWS_DM_00776]	Definition of API function ara::diag::ServiceValidation::Offer
[SWS_DM_00789]	Definition of API function ara::diag::DownloadService::RequestDownload
[SWS_DM_00790]	Definition of API function ara::diag::DownloadService::DownloadData
[SWS_DM_00791]	Definition of API function ara::diag::DownloadService::RequestDownloadExit
[SWS_DM_00792]	Definition of API function ara::diag::DownloadService::Offer
[SWS_DM_00795]	Definition of API class ara::diag::UploadService::OperationOutput
[SWS_DM_00799]	Definition of API function ara::diag::UploadService::RequestUpload
[SWS_DM_00800]	Definition of API function ara::diag::UploadService::UploadData
[SWS_DM_00801]	Definition of API function ara::diag::UploadService::RequestUploadExit
[SWS_DM_00802]	Definition of API function ara::diag::UploadService::Offer
[SWS_DM_00808]	Definition of API function ara::diag::CommunicationControl::CommCtrl Request
[SWS_DM_00809]	Definition of API function ara::diag::CommunicationControl::Offer
[SWS_DM_00836]	Definition of API function ara::diag::DoIPActivationLine::Offer
[SWS_DM_00877]	Starting time-based event debouncing towards `kFinallyDefective`
[SWS_DM_00878]	Starting time-based event debouncing towards `kFinallyHealed`
[SWS_DM_00880]	Debounce time freeze request
[SWS_DM_00921]	Configuration of Error Memory Overflow Indication as extended data record
[SWS_DM_00950]	Configuration of DTC priority as extended data record
[SWS_DM_00951]	Configuration of DTC "current `FDC`" as extended data record
[SWS_DM_00952]	Configuration of DTC "max. `FDC` since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. `FDC` current cycle" as extended data record
[SWS_DM_00954]	Configuration of DTC "occurrence counter" as extended data record
[SWS_DM_00955]	Configuration of DTC "aging counter up/down" as extended data record

▽

△

Number	Heading
[SWS_DM_00958]	Default value for DTC "aging counter up" if aging is not allowed
[SWS_DM_00959]	Default value for DTC "aging counter down" if aging is not allowed
[SWS_DM_00961]	Configuration of a DTCs significance as extended data record
[SWS_DM_00962]	Configuration of a DTCs Failed Operation Cycles as extended data record
[SWS_DM_00963]	Configuration of a DTCs failed operation Cycles Since First Failed as extended data record
[SWS_DM_00964]	Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record
[SWS_DM_00975]	Definition of API function ara::diag::MetaInfo::operator=
[SWS_DM_00976]	Definition of API function ara::diag::MetaInfo::operator=
[SWS_DM_00980]	Definition of API function ara::diag::MetaInfo::~MetaInfo
[SWS_DM_00981]	Conditions of status based reporting order
[SWS_DM_01012]	Definition of API function ara::diag::EcuResetRequest::EnableRapid Shutdown
[SWS_DM_01013]	Definition of API function ara::diag::EcuResetRequest::RequestReset
[SWS_DM_01016]	Definition of API function ara::diag::EcuResetRequest::Offer
[SWS_DM_01081]	Session check for sourceDataIdentifier
[SWS_DM_01082]	Security level check for sourceDataIdentifier
[SWS_DM_01094]	Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On
[SWS_DM_01095]	Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off
[SWS_DM_01126]	Definition of API function ara::diag::Authentication::VerifyCertificate Unidirectional
[SWS_DM_01127]	Definition of API function ara::diag::Authentication::VerifyCertificate Bidirectional
[SWS_DM_01128]	Definition of API function ara::diag::Authentication::VerifyOwnership
[SWS_DM_01129]	Definition of API function ara::diag::Authentication::TransmitCertificate
[SWS_DM_01206]	Set AuthenticationRole
[SWS_DM_01210]	DeAuthenticate due to client inactivity
[SWS_DM_01211]	Transition to DeAuthenticated state on S3server timeout
[SWS_DM_01223]	Diagnostic service role verification
[SWS_DM_01224]	Diagnostic service dynamic access-rights verification
[SWS_DM_01257]	ResourceTemporarilyNotAvailable NRC handling
[SWS_DM_01319]	Consecutive registration of notifier with ReleaseHandler::SetNotifier()
[SWS_DM_01331]	Definition of API function ara::diag::FileTransferService::RequestReadFile
[SWS_DM_01332]	Definition of API function ara::diag::FileTransferService::RequestRead Directory
[SWS_DM_01333]	Definition of API function ara::diag::FileTransferService::RequestWriteFile
[SWS_DM_01334]	Definition of API function ara::diag::FileTransferService::RequestResume WriteFile

▽

△

Number	Heading
[SWS_DM_01335]	Definition of API function ara::diag::FileTransferService::DeleteFile
[SWS_DM_01336]	Definition of API function ara::diag::FileTransferService::Offer
[SWS_DM_01337]	Definition of API function ara::diag::FileTransferService::StopOffer
[SWS_DM_01367]	Definition of API function ara::diag::DoIPEntityIdentification::Offer
[SWS_DM_01375]	Behavior on locked SOVD
[SWS_DM_01376]	Response behavior of SOVD services without access rights
[SWS_DM_01389]	SOVD method Retrieve List of All Supported Modes of an Entity
[SWS_DM_01390]	SOVD operations request and response parameters
[SWS_DM_01404]	SOVD method Retrieve List of All Available Operations from an Entity
[SWS_DM_01405]	SOVD operations
[SWS_DM_01407]	SOVD method Delete All Faults of an Entity without scope
[SWS_DM_01415]	SOVD fault general attributes
[SWS_DM_01417]	SOVD method Read Faults from an Entity
[SWS_DM_01418]	SOVD faults
[SWS_DM_01420]	SOVD data-list
[SWS_DM_01422]	SOVD method Write Configuration as Parameters
[SWS_DM_01424]	SOVD method Read Configuration as Parameters
[SWS_DM_01427]	SOVD configuration attribute name
[SWS_DM_01428]	SOVD configuration attribute id
[SWS_DM_01429]	SOVD method Retrieve List of All Configurations Provided by the Entity
[SWS_DM_01431]	SOVD method Write a Data Value to an Entity
[SWS_DM_01433]	SOVD method Read Single Data Value from an Entity
[SWS_DM_01439]	SOVD data attribute category
[SWS_DM_01440]	SOVD data attribute name
[SWS_DM_01441]	SOVD data attribute id
[SWS_DM_01442]	SOVD method Retrieve List of All Data Provided by the Entity
[SWS_DM_01443]	SOVD method for locking
[SWS_DM_01444]	Data-groups type content
[SWS_DM_01445]	Data-groups type
[SWS_DM_01446]	SOVD method Retrieve Groups Supported by a Data Resource Collection
[SWS_DM_01447]	Data categories type
[SWS_DM_01448]	Standard resource Data categories
[SWS_DM_01449]	SOVD method Retrieve Categories Supported by a Data Resource Collection
[SWS_DM_01469]	Validity period of authenticated roles

▽

△

Number	Heading
[SWS_DM_01470]	Authorization validation
[SWS_DM_01471]	Redirection to token endpoint
[SWS_DM_01472]	Redirection to authorization endpoint
[SWS_DM_01473]	DM shall lock SOVD entity after time expiration
[SWS_DM_01474]	DM shall allow access only to unlocked SOVD entities
[SWS_DM_01475]	DM shall allow only one lock per SOVD entity
[SWS_DM_01476]	Parallel SOVD client handling
[SWS_DM_01477]	SOVD lock in UDS extended session
[SWS_DM_01504]	Definition of API function ara::diag::DataTransferReadSharedData Handler::Read
[SWS_DM_01505]	Definition of API function ara::diag::DataTransferReadSharedData Handler::ExitRead
[SWS_DM_01513]	Definition of API function ara::diag::DataTransferReadByPullHandler::Read
[SWS_DM_01514]	Definition of API function ara::diag::DataTransferReadByPullHandler::Exit Read
[SWS_DM_01522]	Definition of API function ara::diag::DataTransferReadByPushHandler::Read
[SWS_DM_01523]	Definition of API function ara::diag::DataTransferReadByPushHandler::Exit Read
[SWS_DM_01546]	Definition of API function ara::diag::DataTransferWriteHandler::Write
[SWS_DM_01547]	Definition of API function ara::diag::DataTransferWriteHandler::ExitWrite

Table E.26: Changed Specification Items in R23-11

E.9.3 Deleted Specification Items in R23-11

Number	Heading
[SWS_DM_00005]	DoIP Support
[SWS_DM_00040]	Definition of internal debounce counter reset
[SWS_DM_00221]	Handling indicator status
[SWS_DM_00242]	Re-occurrence after Aging
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00879]	Application resetting the internal debounce counter
[SWS_DM_01232]	Handling unspecified negative return values of ara::diag::Authentication::VerifyCertificateUnidirectional method
[SWS_DM_01237]	Handling unspecified negative return values of ara::diag::Authentication::VerifyCertificateBidirectional method
[SWS_DM_01242]	Handling unspecified negative return values of ara::diag::Authentication::VerifyOwnership method

▽

△

Number	Heading
[SWS_DM_01250]	Handling unspecified negative return values of `ara::diag::Authentication::TransmitCertificate` method
[SWS_DM_01438]	`SOVD` data attribute `group`

Table E.27: Deleted Specification Items in R23-11

E.9.4 Added Constraints in R23-11

none

E.9.5 Changed Constraints in R23-11

Number	Heading
[SWS_DM_-CONSTR_-00084]	Each `DTC` shall be assigned to an event memory destination
[SWS_DM_-CONSTR_-00960]	No support for `DEM_AGINGCTR_UPCNT_FIRST_ACTIVE`

Table E.28: Changed Constraints in R23-11

E.9.6 Deleted Constraints in R23-11

Number	Heading
[SWS_DM_-CONSTR_-00208]	Delay time value for sharedTimer

Table E.29: Deleted Constraints in R23-11

E.10 Constraint and Specification Item History of this document according to AUTOSAR Release R24-11

E.10.1 Added Specification Items in R24-11

Number	Heading
[SWS_DM_01951]	No session checks if no diagnostic session inside the diagnostic access permission
[SWS_DM_01952]	No security level checks if no securityLevel inside the diagnostic access permission
[SWS_DM_01953]	`Aging` for untested cycles
[SWS_DM_01954]	Configuration of a DTCs failed operation Cycles Tested Since Last Failed as extended data record
[SWS_DM_01955]	Configuration of a DTCs failed operation Cycles Tested Since First Failed as extended data record
[SWS_DM_01956]	Latching of Internal Data Element `DEM_CYCLES_TESTED_SINCE_LAST_FAILED`
[SWS_DM_01957]	Latching of Internal Data Element `DEM_CYCLES_TESTED_SINCE_FIRST_FAILED`
[SWS_DM_01958]	Availability of internal data element Internal Data Element `DEM_CYCLES_TESTED_SINCE_LAST_FAILED`
[SWS_DM_01959]	AAvailability of internal data element Internal Data Element `DEM_CYCLES_TESTED_SINCE_FIRST_FAILED`
[SWS_DM_01960]	Identical fault memory behavior
[SWS_DM_01961]	Definition of API class ara::diag::TransmitCertificate
[SWS_DM_01962]	Definition of API function ara::diag::TransmitCertificate::TransmitCertificate
[SWS_DM_01963]	Definition of API function ara::diag::TransmitCertificate::TransmitCertificate
[SWS_DM_01964]	Definition of API function ara::diag::TransmitCertificate::TransmitCertificate
[SWS_DM_01965]	Definition of API function ara::diag::TransmitCertificate::operator=
[SWS_DM_01966]	Definition of API function ara::diag::TransmitCertificate::operator=
[SWS_DM_01967]	Definition of API function ara::diag::TransmitCertificate::~TransmitCertificate
[SWS_DM_01968]	Definition of API function ara::diag::TransmitCertificate::Process
[SWS_DM_01969]	Definition of API function ara::diag::TransmitCertificate::Offer
[SWS_DM_01970]	Definition of API function ara::diag::TransmitCertificate::StopOffer
[SWS_DM_01971]	`FDC` value for monitors without debouncing
[SWS_DM_01972]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01973]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01974]	Indicator reporting kOnDemand
[SWS_DM_01975]	Indicator reporting kOff
[SWS_DM_01976]	Example of a configuration table including combined DTCs
[SWS_DM_01977]	Calculation of UDS status byte

△

Number	Heading
[SWS_DM_01978]	Supported sub function of ResponseonEvent (0x86)
[SWS_DM_01979]	Secure Communication for DoIP using TLS
[SWS_DM_01980]	Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration
[SWS_DM_01981]	Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration
[SWS_DM_01982]	Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration
[SWS_DM_01983]	Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration
[SWS_DM_01984]	Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration
[SWS_DM_01985]	Default value for the attribute maxTesterConnections of meta-class DoIpNetworkConfiguration
[SWS_DM_01986]	Used DoIP Protocol Version
[SWS_DM_01987]	Supported debouncing method for which monitor action
[SWS_DM_01988]	Unexpected monitor action handling
[SWS_DM_02000]	Fault memory addressing by the scope parameter
[SWS_DM_02001]	Default fault memory for SOVD requests without scope parameter
[SWS_DM_02002]	Support of entity status response item Max. datasize (MDS)
[SWS_DM_02003]	Behavior of not configured SoftwareClusterDiagnosticDeploymentProps. maxConversations
[SWS_DM_02004]	UDS request priority handling
[SWS_DM_02005]	Providing VIN if interface is not available
[SWS_DM_02006]	Providing EID using manually configured value
[SWS_DM_02007]	Providing EID using MAC of network interface
[SWS_DM_02008]	Providing the GID if application interface not available
[SWS_DM_02009]	Providing the GID using configured value
[SWS_DM_02010]	Providing VIN/GID status byte
[SWS_DM_02011]	VIN/GID status on successful VIN retrieval
[SWS_DM_02012]	VIN/GID status on unsuccessful VIN and GID synchronization
[SWS_DM_02013]	VIN/GID status on unsuccessful VIN and successful GID synchronization
[SWS_DM_02014]	Security events for Diagnostic Management
[SWS_DM_02015]	Reporting security access denied to IdsM
[SWS_DM_02016]	Security event context data definition: SEV_UDS_SECURITY_ACCESS_ NEEDED
[SWS_DM_02017]	Reporting authentication required to IdsM
[SWS_DM_02018]	Security event context data definition: SEV_UDS_AUTHENTICATION_ NEEDED
[SWS_DM_02019]	Reporting successful security access to IdsM

▽

△

Number	Heading
[SWS_DM_02020]	Security event context data definition: SEV_UDS_SECURITY_ACCESS_SUCCESSFUL
[SWS_DM_02021]	Reporting failed security access to IdsM
[SWS_DM_02022]	Security event context data definition: SEV_UDS_SECURITY_ACCESS_FAILED
[SWS_DM_02023]	Reporting successful authentication to IdsM
[SWS_DM_02024]	Security event context data definition: SEV_UDS_AUTHENTICATION_SUCCESSFUL
[SWS_DM_02025]	Reporting failed authentication to IdsM
[SWS_DM_02026]	Security event context data definition: SEV_UDS_AUTHENTICATION_FAILED
[SWS_DM_02027]	Reporting successful writing of data to IdsM
[SWS_DM_02028]	Security event context data definition: SEV_UDS_WRITE_DATA_SUCCESSFUL
[SWS_DM_02029]	Reporting writing of data failed to IdsM
[SWS_DM_02030]	Security event context data definition: SEV_UDS_WRITE_DATA_FAILED
[SWS_DM_02031]	Reporting successful up download to IdsM
[SWS_DM_02032]	Security event context data definition: SEV_UDS_REQUEST_UP_DOWNLOAD_SUCCESSFUL
[SWS_DM_02033]	Reporting up download failed to IdsM
[SWS_DM_02034]	Security event context data definition: SEV_UDS_REQUEST_UP_DOWNLOAD_FAILED
[SWS_DM_02035]	Reporting successful file transfer to IdsM
[SWS_DM_02036]	Security event context data definition: SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL
[SWS_DM_02037]	Reporting file transfer failed to IdsM
[SWS_DM_02038]	Security event context data definition: SEV_UDS_REQUEST_FILE_TRANSFER_FAILED
[SWS_DM_02039]	Reporting successful communication control to IdsM
[SWS_DM_02040]	Security event context data definition: SEV_UDS_COMMUNICATION_CONTROL_SUCCESSFUL
[SWS_DM_02041]	Reporting communication control failed to IdsM
[SWS_DM_02042]	Security event context data definition: SEV_UDS_COMMUNICATION_CONTROL_FAILED
[SWS_DM_02043]	Reporting successful clearing of dtc to IdsM
[SWS_DM_02044]	Security event context data definition: SEV_UDS_CLEAR_DTC_SUCCESSFUL
[SWS_DM_02045]	Reporting clearing of dtc failed to IdsM
[SWS_DM_02046]	Security event context data definition: SEV_UDS_CLEAR_DTC_FAILED
[SWS_DM_02047]	Reporting successful control of dtc to IdsM
[SWS_DM_02048]	Security event context data definition: SEV_UDS_CONTROL_DTC_SETTING_SUCCESSFUL
[SWS_DM_02049]	Reporting control of dtc failed to IdsM

▽

△

Number	Heading
[SWS_DM_02050]	Security event context data definition: SEV_UDS_CONTROL_DTC_SETTING_FAILED
[SWS_DM_02051]	Reporting successful reset of ECU to IdsM
[SWS_DM_02052]	Security event context data definition: SEV_UDS_ECU_RESET_SUCCESSFUL
[SWS_DM_02053]	Reporting reset of ECU failed to IdsM
[SWS_DM_02054]	Security event context data definition: SEV_UDS_ECU_RESET_FAILED
[SWS_DM_02055]	Reporting successful routine control to IdsM
[SWS_DM_02056]	Security event context data definition: SEV_UDS_ROUTINE_CONTROL_SUCCESSFUL
[SWS_DM_02057]	Reporting routine control failed to IdsM
[SWS_DM_02058]	Security event context data definition: SEV_UDS_ROUTINE_CONTROL_FAILED
[SWS_DM_02059]	Derive NRC from DiagUdsNrcErrc
[SWS_DM_02060]	Reaction on ApplicationError
[SWS_DM_02063]	Definition of API function ara::diag::MultipleMonitor::operator=
[SWS_DM_02064]	Definition of API function ara::diag::MultipleMonitor::operator=
[SWS_DM_02065]	Definition of API function ara::diag::MultipleMonitor::MultipleMonitor
[SWS_DM_02066]	Definition of API function ara::diag::MultipleMonitor::MultipleMonitor
[SWS_DM_02067]	Definition of API function ara::diag::MultipleEvent::operator=
[SWS_DM_02068]	Definition of API function ara::diag::MultipleEvent::operator=
[SWS_DM_02069]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_02070]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_02071]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_02072]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_02073]	Definition of API function ara::diag::MultipleCondition::operator=
[SWS_DM_02074]	Definition of API function ara::diag::MultipleCondition::operator=
[SWS_DM_02075]	Definition of API type ara::diag::CancellationHandler::CancellationHandler SetNotifier
[SWS_DM_02076]	Definition of API type ara::diag::DTCInformation::EventMemoryOverflowSet Notifier
[SWS_DM_02077]	Definition of API type ara::diag::ClientAuthentication::ClientAuthenticationSet Notifier
[SWS_DM_02078]	Definition of API type ara::diag::OperationCycle::OperationCycleSetNotifier
[SWS_DM_02079]	Definition of API type ara::diag::ReleaseHandler::ReleaseHandlerSetNotifier
[SWS_DM_02080]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::IsFailedAndTested
[SWS_DM_02081]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::IsPassedAndTested
[SWS_DM_02082]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::IsSet

▽

△

Number	Heading
[SWS_DM_02083]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::IsNotSet
[SWS_DM_02084]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::UdsDtcStatusByteType
[SWS_DM_02085]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::UdsDtcStatusByteType
[SWS_DM_02086]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::UdsDtcStatusByteType
[SWS_DM_02087]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::operator=
[SWS_DM_02088]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByte Type::operator=
[SWS_DM_02089]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol PeriodicHandler::~UdsTransportProtocolPeriodicHandler
[SWS_DM_02100]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::Do IPTriggerVehicleAnnouncement
[SWS_DM_02101]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::~Do IPTriggerVehicleAnnouncement
[SWS_DM_80001]	`Diagnostic Routine Header File`: file name, includes and multiple inclusion guard
[SWS_DM_80002]	`Diagnostic Routine Header File`: service namespace
[SWS_DM_80003]	Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-start-out-arg-symbol>}
[SWS_DM_80004]	Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-stop-out-arg-symbol>}
[SWS_DM_80005]	Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-request-result-out-arg-symbol>}
[SWS_DM_80011]	`Diagnostic DataIdentifier Header File`: file name, includes and multiple inclusion guard
[SWS_DM_80012]	`Diagnostic DataIdentifier Header File`: service namespace
[SWS_DM_80013]	Definition of API variable {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-out-arg-symbol>}
[SWS_DM_80021]	`Diagnostic DataElement Header File`: file name, includes and multiple inclusion guard
[SWS_DM_80022]	`Diagnostic DataElement Header File`: service namespace

▽

△

Number	Heading
[SWS_DM_80023]	Definition of API variable {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-out-arg-symbol>}

Table E.30: Added Specification Items in R24-11

E.10.2 Changed Specification Items in R24-11

Number	Heading
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00017]	Calculation of the FDC based on the internal debounce counter
[SWS_DM_00018]	Internal debounce counter init
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00022]	Internal debounce counter jump up behavior
[SWS_DM_00023]	Internal debounce counter jump down behavior
[SWS_DM_00024]	Qualified failed event using counter-based debouncing
[SWS_DM_00025]	Qualified passed event using counter-based debouncing
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00033]	Debounce timer behavior upon reported kFailed
[SWS_DM_00036]	Debounce timer behavior upon reported kPassed
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00039]	Resetting the internal debounce counter upon restarting an operation cycle
[SWS_DM_00049]	Refusal of diagnostic request due to prioritization with BusyRepeatRequest
[SWS_DM_00055]	Supported event memories
[SWS_DM_00056]	Availability of the primary event memory
[SWS_DM_00057]	Availability of a user-defined event memory
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00060]	Set of supported DTCs
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response Read DTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1 and Autosar Diagnostic Extract Template of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response Read DTCInformation.reportNumberOfDTCBySeverityMaskRecord
[SWS_DM_00064]	Definition of DTC groups
[SWS_DM_00065]	Always supported availability of the group of all DTCs
[SWS_DM_00083]	Event memory destination of an DTC

▽

△

Number	Heading
[SWS_DM_00085]	Internal debounce timer init
[SWS_DM_00086]	Resetting the internal debounce counter after clearing DTC
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00091]	Evaluation of ClearDiagnosticInformation parameters
[SWS_DM_00092]	Parameter range check for groupOfDTC request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00097]	Abort on failed verification step
[SWS_DM_00098]	UDS message checks
[SWS_DM_00099]	Supported Service SID level checks
[SWS_DM_00100]	Supported Service subfunction level checks
[SWS_DM_00101]	Session Access SID level Permission
[SWS_DM_00102]	Session Access subfunction level Permission
[SWS_DM_00103]	Security Access level Permission
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service 0x14
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00116]	Clearing a DTC group
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00124]	Limited clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00126]	Realisation of UDS service 0x3E TesterPresent
[SWS_DM_00127]	Availability of diagnostic service processors
[SWS_DM_00128]	Realization of UDS service RequestDownload (0x34)
[SWS_DM_00129]	Supported addressAndLengthFormatIdentifier
[SWS_DM_00130]	Not supported addressAndLengthFormatIdentifier
[SWS_DM_00134]	Realization of UDS service RequestUpload (0x35)
[SWS_DM_00137]	Realization of UDS service TransferData (0x36)
[SWS_DM_00140]	Realisation of UDS service 0x28 CommunicationControl
[SWS_DM_00141]	Realization of UDS service RequestTransferExit (0x37)
[SWS_DM_00144]	Parallel clearing DTCs in different DiagnosticMemoryDestination
[SWS_DM_00145]	Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination
[SWS_DM_00146]	Unlock clear DTC operation for one DiagnosticMemoryDestination
[SWS_DM_00147]	Behavior while trying to clear DTCs on a locked DiagnosticMemoryDestination

▽

△

Number	Heading
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00150]	Primary trigger for event memory entry storage
[SWS_DM_00151]	snapshot record numeration
[SWS_DM_00152]	Number of snapshot records for a DTC
[SWS_DM_00154]	Number of extended data for a DTC
[SWS_DM_00155]	Extended data record numeration
[SWS_DM_00159]	Allow only to clear GroupOfAllDTCs
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00162]	Point in time for positive response for ClearDTC
[SWS_DM_00163]	Definition of a inhibited clear operation on single DTC
[SWS_DM_00164]	Definition of a inhibited clear operation for a group of DTCs
[SWS_DM_00170]	Realisation of UDS service ReadDataByIdentifier (0x22)
[SWS_DM_00186]	Realisation of UDS service WriteDataByIdentifier (0x2E)
[SWS_DM_00193]	Support of a user-defined fault memory clear request
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnostic Information
[SWS_DM_00195]	Clearing a user-defined memory
[SWS_DM_00199]	Positive Response processing
[SWS_DM_00201]	Realization of UDS service RoutineControl (0x31)
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00208]	Validation of the requested user-defined memory number
[SWS_DM_00213]	DTC status processing
[SWS_DM_00217]	UDS DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	kConfirmedDTC (Bit3) calculation
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing
[SWS_DM_00226]	Support of UDS service DiagnosticSessionControl
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00228]	Switch to requested Diagnostic Session
[SWS_DM_00229]	Support of UDS service ControlDTCSetting (0x85)
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00238]	Aging and healing
[SWS_DM_00239]	Aging counter

▽

△

Number	Heading
[SWS_DM_00240]	Processing the `Aging` counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00243]	`Aging`-related `UDS DTC status byte` processing
[SWS_DM_00244]	Support of `UDS` service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of `UDS` service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of `UDS` service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of `UDS` service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00252]	Reaction on Unsupported Subfunction
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00277]	Cancellation of a Diagnostic Conversation in case of External Service Processing
[SWS_DM_00278]	Cancellation of a Diagnostic Conversation in case of Internal Processing
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00287]	Configurable environmental condition check criteria
[SWS_DM_00288]	Configurable environmental condition check evaluates to `TRUE`
[SWS_DM_00289]	Configurable environmental condition check evaluates to `FALSE`
[SWS_DM_00290]	Refusal of diagnostic request due to prioritization without response
[SWS_DM_00291]	Definition of API class apext::diag::uds_transport::UdsMessage
[SWS_DM_00293]	Definition of API type apext::diag::uds_transport::UdsMessage::Address
[SWS_DM_00294]	Definition of API type apext::diag::uds_transport::UdsMessage::MetaInfo Map
[SWS_DM_00296]	Definition of API enum apext::diag::uds_transport::UdsMessage::Target AddressType
[SWS_DM_00297]	Definition of API function apext::diag::uds_transport::UdsMessage::GetSa
[SWS_DM_00298]	Definition of API function apext::diag::uds_transport::UdsMessage::GetTa
[SWS_DM_00299]	Definition of API function apext::diag::uds_transport::UdsMessage::GetTa Type
[SWS_DM_00300]	Definition of API function apext::diag::uds_transport::UdsMessage::Get Payload
[SWS_DM_00301]	Definition of API function apext::diag::uds_transport::UdsMessage::Get Payload
[SWS_DM_00302]	Definition of API function apext::diag::uds_transport::UdsMessage::AddMeta Info
[SWS_DM_00303]	Definition of API type apext::diag::uds_transport::UdsMessagePtr

▽

△

Number	Heading
[SWS_DM_00304]	Definition of API type apext::diag::uds_transport::UdsMessageConstPtr
[SWS_DM_00306]	Definition of API class apext::diag::uds_transport::UdsTransportProtocolMgr
[SWS_DM_00307]	Definition of API enum apext::diag::uds_transport::UdsTransportProtocol Mgr::TransmissionResult
[SWS_DM_00309]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::IndicateMessage
[SWS_DM_00310]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::NotifyMessageFailure
[SWS_DM_00311]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::HandleMessage
[SWS_DM_00312]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::TransmitConfirmation
[SWS_DM_00313]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::ChannelReestablished
[SWS_DM_00314]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::HandlerStopped
[SWS_DM_00315]	Definition of API class apext::diag::uds_transport::UdsTransportProtocol Handler
[SWS_DM_00319]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::Initialize
[SWS_DM_00322]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::Start
[SWS_DM_00323]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::Stop
[SWS_DM_00325]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::GetHandlerID
[SWS_DM_00326]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::NotifyReestablishment
[SWS_DM_00327]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::Transmit
[SWS_DM_00329]	Lifecycle management of an UDS Transport Protocol implementation
[SWS_DM_00330]	Construction of an UDS Transport Protocol implementation
[SWS_DM_00331]	Initialization of an UDS Transport Protocol implementation
[SWS_DM_00332]	Starting of an UDS Transport Protocol implementation
[SWS_DM_00333]	Stopping of an UDS Transport Protocol implementation
[SWS_DM_00336]	Definition of API type apext::diag::uds_transport::UdsTransportProtocol HandlerID
[SWS_DM_00337]	Definition of API type apext::diag::uds_transport::ChannelID
[SWS_DM_00338]	Definition of API type apext::diag::uds_transport::ByteSpan
[SWS_DM_00340]	Waiting for Stop confirmation
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message

▽

△

Number	Heading
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00363]	Unsupported Subfunction
[SWS_DM_00368]	DM takes care of Response Pending Messages
[SWS_DM_00369]	Maximum number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_00378]	ControlDTCSetting influence
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00382]	Session timeout start
[SWS_DM_00383]	Session timeout stop
[SWS_DM_00384]	Definition of API enum apext::diag::uds_transport::UdsTransportProtocol Mgr::IndicationResult
[SWS_DM_00385]	Acceptance of UDS message reception
[SWS_DM_00386]	Ignoring UDS message reception because DM is busy
[SWS_DM_00387]	Ignoring UDS message reception because DM has no (memory) ressources
[SWS_DM_00388]	Filling provided UdsMessage
[SWS_DM_00389]	Skipping Forwarding of UDS message
[SWS_DM_00390]	Dispatching physical Request
[SWS_DM_00391]	Dispatching functional Request
[SWS_DM_00392]	Properties of returned UdsMessage
[SWS_DM_00393]	Retrieving data for internal DiagnosticDataElements
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00409]	Check supported DataIdentifier
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level

▽

△

Number	Heading
[SWS_DM_00415]	Check supported DataIdentifier
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00421]	Identification of a Diagnostic Client
[SWS_DM_00426]	Assigning a UDS request to an existing Diagnostic Conversation
[SWS_DM_00428]	Treatment of priority values
[SWS_DM_00429]	Prioritization in active non-default session
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations in active default session
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[SWS_DM_00433]	Refusal of diagnostic request due to busy Diagnostic Conversation
[SWS_DM_00437]	Security Level check for RoutineIdentifier
[SWS_DM_00448]	Check supported RoutineIdentifier in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00450]	Security Access subfunction level Permission
[SWS_DM_00475]	Support of DoIP based on ISO 13400-2
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Security Access Delay Timer on power up when attempt counter threshold is reached
[SWS_DM_00480]	Shared Security Access Delay Timer
[SWS_DM_00487]	Ignoring UDS message reception because of unknown target address
[SWS_DM_00491]	Realisation of UDS service 0x86 ResponseOnEvent
[SWS_DM_00493]	Reestablishing of Client Server communication
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00502]	Support for Custom Diagnostic Services
[SWS_DM_00507]	Length check on UDS Service 0x27 request with Subfunction for Request Seed
[SWS_DM_00514]	Definition of API enum ara::diag::DiagErrc
[SWS_DM_00515]	Definition of API class ara::diag::DiagException
[SWS_DM_00516]	Definition of API function ara::diag::DiagException::DiagException
[SWS_DM_00517]	Definition of API class ara::diag::DiagErrorDomain
[SWS_DM_00518]	Definition of API type ara::diag::DiagErrorDomain::Errc
[SWS_DM_00519]	Definition of API type ara::diag::DiagErrorDomain::Exception
[SWS_DM_00520]	Definition of API function ara::diag::DiagErrorDomain::DiagErrorDomain
[SWS_DM_00521]	Definition of API function ara::diag::DiagErrorDomain::Name
[SWS_DM_00522]	Definition of API function ara::diag::DiagErrorDomain::Message
[SWS_DM_00523]	Definition of API function ara::diag::DiagErrorDomain::ThrowAsException
[SWS_DM_00524]	Definition of API function ara::diag::GetDiagDomain
[SWS_DM_00525]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_00526]	Definition of API enum ara::diag::DiagUdsNrcErrc

▽

△

Number	Heading
[SWS_DM_00527]	Definition of API class ara::diag::DiagUdsNrcException
[SWS_DM_00528]	Definition of API function ara::diag::DiagUdsNrcException::DiagUdsNrc Exception
[SWS_DM_00529]	Definition of API class ara::diag::DiagUdsNrcErrorDomain
[SWS_DM_00530]	Definition of API type ara::diag::DiagUdsNrcErrorDomain::Errc
[SWS_DM_00531]	Definition of API type ara::diag::DiagUdsNrcErrorDomain::Exception
[SWS_DM_00532]	Definition of API function ara::diag::DiagUdsNrcErrorDomain::DiagUdsNrc ErrorDomain
[SWS_DM_00533]	Definition of API function ara::diag::DiagUdsNrcErrorDomain::Name
[SWS_DM_00534]	Definition of API function ara::diag::DiagUdsNrcErrorDomain::Message
[SWS_DM_00535]	Definition of API function ara::diag::DiagUdsNrcErrorDomain::ThrowAs Exception
[SWS_DM_00536]	Definition of API function ara::diag::GetDiagUdsNrcDomain
[SWS_DM_00537]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_00538]	Definition of API class ara::diag::CounterBased
[SWS_DM_00539]	Definition of API class ara::diag::TimeBased
[SWS_DM_00540]	Definition of API enum ara::diag::InitMonitorReason
[SWS_DM_00541]	Definition of API enum ara::diag::MonitorAction
[SWS_DM_00542]	Definition of API class ara::diag::Monitor
[SWS_DM_00543]	Definition of API function ara::diag::Monitor::ReportMonitorAction
[SWS_DM_00548]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_00549]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_00550]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_00551]	Definition of API class ara::diag::GenericRoutine::OperationOutput
[SWS_DM_00552]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_00553]	Definition of API function ara::diag::GenericRoutine::~GenericRoutine
[SWS_DM_00554]	Definition of API function ara::diag::GenericRoutine::Start
[SWS_DM_00555]	Definition of API function ara::diag::GenericRoutine::Stop
[SWS_DM_00556]	Definition of API function ara::diag::GenericRoutine::RequestResults
[SWS_DM_00557]	Definition of API function ara::diag::GenericRoutine::Offer
[SWS_DM_00558]	Definition of API function ara::diag::GenericRoutine::StopOffer
[SWS_DM_00559]	Definition of API enum ara::diag::DiagOfferErrc
[SWS_DM_00560]	Definition of API enum ara::diag::DiagReportingErrc
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00568]	Handling of `enable conditions`
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing with generic interface

▽

△

Number	Heading
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing with generic interface
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing with generic interface
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00578]	Definition of API class ara::diag::GenericUDSService::OperationOutput
[SWS_DM_00579]	Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-name-upper-camel>}Read
[SWS_DM_00580]	Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-name-upper-camel>}Output
[SWS_DM_00581]	Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}StartOutput
[SWS_DM_00582]	Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}StopOutput
[SWS_DM_00583]	Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name-upper-camel>}RequestResultOutput
[SWS_DM_00584]	Definition of API function ara::diag::GenericUDSService::~GenericUDSService
[SWS_DM_00585]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}
[SWS_DM_00586]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::~{<data-identifier-interface-name>}
[SWS_DM_00587]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::{<data-element-interface-name>}
[SWS_DM_00588]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::~{<data-element-interface-name>}
[SWS_DM_00589]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}
[SWS_DM_00590]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::~{<routine-interface-name>}

▽

△

Number	Heading
[SWS_DM_00591]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Start
[SWS_DM_00592]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Stop
[SWS_DM_00593]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::RequestResult
[SWS_DM_00594]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::Offer
[SWS_DM_00595]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::StopOffer
[SWS_DM_00596]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::Read
[SWS_DM_00597]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name>}::Offer
[SWS_DM_00598]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::Write
[SWS_DM_00599]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::Offer
[SWS_DM_00600]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::StopOffer
[SWS_DM_00601]	Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}
[SWS_DM_00602]	Definition of API class ara::diag::GenericUDSService
[SWS_DM_00603]	Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name>}
[SWS_DM_00604]	Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}
[SWS_DM_00605]	Definition of API class ara::diag::GenericRoutine
[SWS_DM_00607]	Definition of API class ara::diag::GenericDataIdentifier
[SWS_DM_00608]	Definition of API class ara::diag::CancellationHandler
[SWS_DM_00609]	Definition of API function ara::diag::CancellationHandler::CancellationHandler
[SWS_DM_00610]	Definition of API function ara::diag::CancellationHandler::CancellationHandler
[SWS_DM_00611]	Definition of API function ara::diag::CancellationHandler::CancellationHandler
[SWS_DM_00612]	Definition of API function ara::diag::CancellationHandler::operator=
[SWS_DM_00613]	Definition of API function ara::diag::CancellationHandler::operator=
[SWS_DM_00614]	Definition of API function ara::diag::CancellationHandler::IsCanceled

▽

△

Number	Heading
[SWS_DM_00615]	Definition of API function ara::diag::CancellationHandler::SetNotifier
[SWS_DM_00616]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_00617]	Definition of API function {<namespace-list-data-element>}::{ <data-element-interface-name>}::StopOffer
[SWS_DM_00618]	Definition of API function ara::diag::GenericUDSService::HandleMessage
[SWS_DM_00619]	Definition of API function ara::diag::GenericUDSService::Offer
[SWS_DM_00620]	Definition of API function ara::diag::GenericUDSService::StopOffer
[SWS_DM_00621]	Definition of API variable ara::diag::CounterBased::failedThreshold
[SWS_DM_00622]	Definition of API variable ara::diag::CounterBased::passedThreshold
[SWS_DM_00623]	Definition of API variable ara::diag::CounterBased::failedStepsize
[SWS_DM_00624]	Definition of API variable ara::diag::CounterBased::passedStepsize
[SWS_DM_00625]	Definition of API variable ara::diag::CounterBased::failedJumpValue
[SWS_DM_00626]	Definition of API variable ara::diag::CounterBased::passedJumpValue
[SWS_DM_00627]	Definition of API variable ara::diag::CounterBased::useJumpToFailed
[SWS_DM_00628]	Definition of API variable ara::diag::CounterBased::useJumpToPassed
[SWS_DM_00629]	Definition of API variable ara::diag::TimeBased::failedMs
[SWS_DM_00630]	Definition of API variable ara::diag::TimeBased::passedMs
[SWS_DM_00631]	Definition of API variable ara::diag::GenericDataIdentifier::Operation Output::responseData
[SWS_DM_00632]	Definition of API variable ara::diag::GenericUDSService::Operation Output::responseData
[SWS_DM_00633]	Definition of API variable ara::diag::GenericRoutine::Operation Output::responseData
[SWS_DM_00634]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_00635]	Definition of API function ara::diag::GenericDataIdentifier::~GenericData Identifier
[SWS_DM_00636]	Definition of API function ara::diag::GenericDataIdentifier::Read
[SWS_DM_00637]	Definition of API function ara::diag::GenericDataIdentifier::Write
[SWS_DM_00638]	Definition of API function ara::diag::GenericDataIdentifier::Offer
[SWS_DM_00639]	Definition of API function ara::diag::GenericDataIdentifier::StopOffer
[SWS_DM_00640]	Definition of API function {<namespace-list-data-identifier>}::{ <data-identifier-interface-name>}::Read
[SWS_DM_00641]	Definition of API class ara::diag::GenericDataIdentifier::OperationOutput
[SWS_DM_00642]	Definition of API enum ara::diag::DTCFormatType
[SWS_DM_00643]	Definition of API enum ara::diag::EventStatusBit
[SWS_DM_00644]	Definition of API class ara::diag::EventStatusByte
[SWS_DM_00645]	Definition of API enum ara::diag::DebouncingState
[SWS_DM_00646]	Definition of API class ara::diag::Event
[SWS_DM_00647]	Definition of API function ara::diag::Event::Event

▽

△

Number	Heading
[SWS_DM_00648]	Definition of API function ara::diag::Event::~Event
[SWS_DM_00649]	Definition of API function ara::diag::Event::GetEventStatus
[SWS_DM_00650]	Definition of API function ara::diag::Event::SetEventStatusChangedNotifier
[SWS_DM_00651]	Definition of API function ara::diag::Event::GetLatchedWIRStatus
[SWS_DM_00652]	Definition of API function ara::diag::Event::SetLatchedWIRStatus
[SWS_DM_00653]	Definition of API function ara::diag::Event::GetDTCNumber
[SWS_DM_00654]	Definition of API function ara::diag::Event::GetDebouncingStatus
[SWS_DM_00655]	Definition of API function ara::diag::Event::GetTestComplete
[SWS_DM_00656]	Definition of API function ara::diag::Event::GetFaultDetectionCounter
[SWS_DM_00657]	Definition of API class ara::diag::DTCInformation
[SWS_DM_00658]	Definition of API enum ara::diag::UdsDtcStatusBitType
[SWS_DM_00659]	Definition of API class ara::diag::DTCInformation::UdsDtcStatusByteType
[SWS_DM_00660]	Definition of API class ara::diag::DTCInformation::SnapshotDataIdentifier Type
[SWS_DM_00661]	Definition of API class ara::diag::DTCInformation::SnapshotDataRecordType
[SWS_DM_00662]	Definition of API class ara::diag::DTCInformation::SnapshotRecordUpdated Type
[SWS_DM_00663]	Definition of API enum ara::diag::ControlDtcStatusType
[SWS_DM_00664]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_00665]	Definition of API function ara::diag::DTCInformation::~DTCInformation
[SWS_DM_00666]	Definition of API function ara::diag::DTCInformation::GetCurrentStatus
[SWS_DM_00667]	Definition of API function ara::diag::DTCInformation::SetDTCStatusChanged Notifier
[SWS_DM_00668]	Definition of API function ara::diag::DTCInformation::SetSnapshotRecord UpdatedNotifier
[SWS_DM_00669]	Definition of API function ara::diag::DTCInformation::GetNumberOfStored Entries
[SWS_DM_00670]	Definition of API function ara::diag::DTCInformation::SetNumberOfStored EntriesNotifier
[SWS_DM_00671]	Definition of API function ara::diag::DTCInformation::Clear
[SWS_DM_00672]	Definition of API function ara::diag::DTCInformation::GetControlDTCStatus
[SWS_DM_00673]	Definition of API function ara::diag::DTCInformation::SetControlDtcStatus Notifier
[SWS_DM_00674]	Definition of API function ara::diag::DTCInformation::EnableControlDtc
[SWS_DM_00690]	Definition of API enum ara::diag::ActivityStatusType
[SWS_DM_00691]	Definition of API class ara::diag::Conversation::ConversationIdentifierType
[SWS_DM_00692]	Definition of API function ara::diag::Conversation::GetConversation
[SWS_DM_00693]	Definition of API class ara::diag::Conversation
[SWS_DM_00694]	Definition of API function ara::diag::Conversation::GetActivityStatus
[SWS_DM_00695]	Definition of API function ara::diag::Conversation::SetActivityNotifier

▽

△

Number	Heading
[SWS_DM_00696]	Definition of API function ara::diag::Conversation::GetDiagnosticSession
[SWS_DM_00697]	Definition of API function ara::diag::Conversation::SetDiagnosticSession Notifier
[SWS_DM_00698]	Definition of API function ara::diag::Conversation::GetDiagnosticSecurity Level
[SWS_DM_00699]	Definition of API function ara::diag::Conversation::SetSecurityLevelNotifier
[SWS_DM_00700]	Definition of API function ara::diag::Conversation::GetConversationIdentifier
[SWS_DM_00701]	Definition of API function ara::diag::Conversation::ResetToDefaultSession
[SWS_DM_00705]	Definition of API type ara::diag::SecurityLevelType
[SWS_DM_00706]	Definition of API type ara::diag::SessionControlType
[SWS_DM_00707]	Definition of API function ara::diag::Conversation::GetDiagnosticSession ShortName
[SWS_DM_00708]	Definition of API function ara::diag::Conversation::GetDiagnosticSecurity LevelShortName
[SWS_DM_00710]	Definition of API enum ara::diag::ConditionType
[SWS_DM_00711]	Definition of API class ara::diag::Condition
[SWS_DM_00712]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_00713]	Definition of API function ara::diag::Condition::~Condition
[SWS_DM_00714]	Definition of API function ara::diag::Condition::GetCondition
[SWS_DM_00715]	Definition of API function ara::diag::Condition::SetCondition
[SWS_DM_00720]	Definition of API class ara::diag::DoIPGroupIdentification
[SWS_DM_00721]	Definition of API class ara::diag::DoIPGroupIdentification::GidStatus
[SWS_DM_00722]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroup Identification
[SWS_DM_00723]	Definition of API function ara::diag::DoIPGroupIdentification::~DoIPGroup Identification
[SWS_DM_00724]	Definition of API function ara::diag::DoIPGroupIdentification::GetGidStatus
[SWS_DM_00725]	Definition of API function ara::diag::DoIPGroupIdentification::Offer
[SWS_DM_00726]	Definition of API function ara::diag::DoIPGroupIdentification::StopOffer
[SWS_DM_00730]	Definition of API enum ara::diag::PowerModeType
[SWS_DM_00731]	Definition of API class ara::diag::DoIPPowerMode
[SWS_DM_00732]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_00733]	Definition of API function ara::diag::DoIPPowerMode::~DoIPPowerMode
[SWS_DM_00734]	Definition of API function ara::diag::DoIPPowerMode::GetDoIPPowerMode
[SWS_DM_00735]	Definition of API function ara::diag::DoIPPowerMode::Offer
[SWS_DM_00736]	Definition of API function ara::diag::DoIPPowerMode::StopOffer
[SWS_DM_00740]	Definition of API enum ara::diag::IndicatorStatusType
[SWS_DM_00741]	Definition of API class ara::diag::Indicator
[SWS_DM_00742]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_00743]	Definition of API function ara::diag::Indicator::~Indicator

▽

△

Number	Heading
[SWS_DM_00744]	Definition of API function ara::diag::Indicator::GetIndicatorState
[SWS_DM_00745]	Definition of API function ara::diag::Indicator::SetNotifier
[SWS_DM_00751]	Definition of API class ara::diag::OperationCycle
[SWS_DM_00752]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_00753]	Definition of API function ara::diag::OperationCycle::~OperationCycle
[SWS_DM_00755]	Definition of API function ara::diag::OperationCycle::SetNotifier
[SWS_DM_00760]	Definition of API enum ara::diag::KeyCompareResultType
[SWS_DM_00761]	Definition of API class ara::diag::SecurityAccess
[SWS_DM_00762]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_00763]	Definition of API function ara::diag::SecurityAccess::~SecurityAccess
[SWS_DM_00764]	Definition of API function ara::diag::SecurityAccess::GetSeed
[SWS_DM_00765]	Definition of API function ara::diag::SecurityAccess::CompareKey
[SWS_DM_00766]	Definition of API function ara::diag::SecurityAccess::Offer
[SWS_DM_00767]	Definition of API function ara::diag::SecurityAccess::StopOffer
[SWS_DM_00770]	Definition of API enum ara::diag::ConfirmationStatusType
[SWS_DM_00771]	Definition of API class ara::diag::ServiceValidation
[SWS_DM_00772]	Definition of API function ara::diag::ServiceValidation::ServiceValidation
[SWS_DM_00773]	Definition of API function ara::diag::ServiceValidation::~ServiceValidation
[SWS_DM_00774]	Definition of API function ara::diag::ServiceValidation::Validate
[SWS_DM_00775]	Definition of API function ara::diag::ServiceValidation::Confirmation
[SWS_DM_00776]	Definition of API function ara::diag::ServiceValidation::Offer
[SWS_DM_00777]	Definition of API function ara::diag::ServiceValidation::StopOffer
[SWS_DM_00782]	Definition of API function ara::diag::Conversation::GetAllConversations
[SWS_DM_00783]	Definition of API function ara::diag::Conversation::GetCurrentActive Conversations
[SWS_DM_00784]	Definition of API class ara::diag::DownloadService
[SWS_DM_00785]	Definition of API class ara::diag::DownloadService::OperationOutput
[SWS_DM_00786]	Definition of API variable ara::diag::DownloadService::Operation Output::responseData
[SWS_DM_00787]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_00788]	Definition of API function ara::diag::DownloadService::~DownloadService
[SWS_DM_00789]	Definition of API function ara::diag::DownloadService::RequestDownload
[SWS_DM_00790]	Definition of API function ara::diag::DownloadService::DownloadData
[SWS_DM_00791]	Definition of API function ara::diag::DownloadService::RequestDownloadExit
[SWS_DM_00792]	Definition of API function ara::diag::DownloadService::Offer
[SWS_DM_00793]	Definition of API function ara::diag::DownloadService::StopOffer
[SWS_DM_00794]	Definition of API class ara::diag::UploadService
[SWS_DM_00795]	Definition of API class ara::diag::UploadService::OperationOutput

▽

△

Number	Heading
[SWS_DM_00796]	Definition of API variable ara::diag::UploadService::Operation Output::responseData
[SWS_DM_00797]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_00798]	Definition of API function ara::diag::UploadService::~UploadService
[SWS_DM_00799]	Definition of API function ara::diag::UploadService::RequestUpload
[SWS_DM_00800]	Definition of API function ara::diag::UploadService::UploadData
[SWS_DM_00801]	Definition of API function ara::diag::UploadService::RequestUploadExit
[SWS_DM_00802]	Definition of API function ara::diag::UploadService::Offer
[SWS_DM_00803]	Definition of API function ara::diag::UploadService::StopOffer
[SWS_DM_00804]	Definition of API class ara::diag::CommunicationControl
[SWS_DM_00805]	Definition of API class ara::diag::CommunicationControl::ComCtrlRequest ParamsType
[SWS_DM_00806]	Definition of API function ara::diag::CommunicationControl::Communication Control
[SWS_DM_00807]	Definition of API function ara::diag::Communication Control::~CommunicationControl
[SWS_DM_00808]	Definition of API function ara::diag::CommunicationControl::CommCtrl Request
[SWS_DM_00809]	Definition of API function ara::diag::CommunicationControl::Offer
[SWS_DM_00810]	Definition of API function ara::diag::CommunicationControl::StopOffer
[SWS_DM_00811]	Re-enabling of ControlDTCSetting by Diagnostic Application
[SWS_DM_00812]	Re-enabling on transition to default session
[SWS_DM_00813]	Providing the `GID` in `DoIP` protocol messages using application interface
[SWS_DM_00814]	Providing the `PowerMode` in DoIP protocol messages
[SWS_DM_00815]	When to send Vehicle announcement messages on interfaces without activation line control
[SWS_DM_00816]	Notification of activation line status change on activation line controlled network interfaces
[SWS_DM_00820]	Definition of API class ara::diag::DoIPTriggerVehicleAnnouncement
[SWS_DM_00821]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::Get DoIPTriggerVehicleAnnouncement
[SWS_DM_00822]	Definition of API function ara::diag::DoIPTriggerVehicle Announcement::TriggerVehicleAnnouncement
[SWS_DM_00830]	Definition of API class ara::diag::DoIPActivationLine
[SWS_DM_00831]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_00832]	Definition of API function ara::diag::DoIPActivationLine::~DoIPActivationLine
[SWS_DM_00833]	Definition of API function ara::diag::DoIPActivationLine::GetNetworkInterface Id
[SWS_DM_00834]	Definition of API function ara::diag::DoIPActivationLine::UpdateActivation LineState
[SWS_DM_00835]	Definition of API function ara::diag::DoIPActivationLine::GetActivationLine State

▽

△

Number	Heading
[SWS_DM_00836]	Definition of API function ara::diag::DoIPActivationLine::Offer
[SWS_DM_00837]	Definition of API function ara::diag::DoIPActivationLine::StopOffer
[SWS_DM_00840]	Instantiation of Diagnostic Conversation Interface
[SWS_DM_00841]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00842]	Default session change trigger from AAs
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00844]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00845]	Notification about session change
[SWS_DM_00846]	Notification about security-level change
[SWS_DM_00847]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00855]	Providing the VIN in DoIP protocol messages
[SWS_DM_00856]	Initial values for Diagnostic Conversation
[SWS_DM_00859]	Confirmation of service processing
[SWS_DM_00860]	No service processing
[SWS_DM_00863]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00864]	Checking Supported Subfunction for CompareKey
[SWS_DM_00865]	Communication control service processing
[SWS_DM_00866]	Negative Response processing
[SWS_DM_00867]	UDS service RequestDownload (0x34) processing
[SWS_DM_00868]	UDS service RequestUpload (0x35) processing
[SWS_DM_00869]	UDS service TransferData (0x36) download processing
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) download processing
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00877]	Starting time-based event debouncing towards kFinallyDefective
[SWS_DM_00878]	Starting time-based event debouncing towards kFinallyHealed
[SWS_DM_00880]	Debounce time freeze request
[SWS_DM_00882]	Enable condition influence on debouncing behavior
[SWS_DM_00883]	UDS DTC status bit transitions triggered by test results
[SWS_DM_00886]	Observability of the Event status byte
[SWS_DM_00888]	Observability of indicator status
[SWS_DM_00894]	Notification event upon snapshot record updates
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00896]	Handling of DiagnosticClearConditions
[SWS_DM_00897]	Usage of ClearDTC Interface
[SWS_DM_00898]	ClearDTC call on invalid DTC or DTC group

▽

△

Number	Heading
[SWS_DM_00899]	ClearDTC called while another clear operation is in progress
[SWS_DM_00900]	ClearDTC processing in case of memory errors
[SWS_DM_00901]	Possible failure of ClearDTC
[SWS_DM_00902]	NumberOfStoredEntries
[SWS_DM_00905]	Retrieving data for `external DiagnosticDataElements`
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00908]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00909]	Support of Subfunction 0x01 (ON)
[SWS_DM_00910]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00911]	Instances of DTCInformation interface
[SWS_DM_00916]	Priority values
[SWS_DM_00918]	Definition of API function ara::diag::DTCInformation::SetEventMemory OverflowNotifier
[SWS_DM_00919]	Definition of API function ara::diag::DTCInformation::GetEventMemory Overflow
[SWS_DM_00920]	Configuration of the event memory size
[SWS_DM_00921]	Configuration of Error Memory Overflow Indication as extended data record
[SWS_DM_00922]	Persistent storage for event memory overflow information
[SWS_DM_00923]	Event memory overflow set condition
[SWS_DM_00924]	Event memory overflow reset condition
[SWS_DM_00925]	Event memory overflow notifier on occurence
[SWS_DM_00926]	Event memory overflow notifier on clear
[SWS_DM_00927]	Disabled displacement
[SWS_DM_00928]	Priority and occurrence based displacement
[SWS_DM_00929]	Displacement strategy "full"
[SWS_DM_00930]	Displacement operation
[SWS_DM_00932]	`UDS DTC status bit` 3 / 'ConfirmedDTC' after displacement
[SWS_DM_00933]	`UDS DTC status bit` 5 / 'testFailedSinceLastClear' after displacement
[SWS_DM_00934]	Condition for discarding the new event
[SWS_DM_00935]	Definition of API enum ara::diag::ConcurrencyType
[SWS_DM_00936]	Definition of API class ara::diag::DataIdentifierConcurrencyType
[SWS_DM_00937]	Definition of API variable ara::diag::DataIdentifierConcurrencyType::read
[SWS_DM_00938]	Definition of API variable ara::diag::DataIdentifierConcurrencyType::write
[SWS_DM_00939]	Definition of API variable ara::diag::DataIdentifierConcurrencyType::read Write
[SWS_DM_00940]	Concurrent ara::diag interface calls for service processing
[SWS_DM_00941]	Concurrent ara::diag interface calls for DID read processing
[SWS_DM_00942]	Concurrent ara::diag interface calls for DID write processing
[SWS_DM_00943]	Concurrent ara::diag interface calls for DID read and write processing

▽

△

Number	Heading
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00945]	Occurrence Counter initial value
[SWS_DM_00946]	Occurrence Counter increment strategy 'testFailed'-only
[SWS_DM_00947]	Occurrence Counter increment strategy 'confirmedDtcBit'
[SWS_DM_00948]	Occurrence Counter upper limit
[SWS_DM_00949]	Generation and usage of internal DiagnosticDataElements
[SWS_DM_00950]	Configuration of DTC priority as extended data record
[SWS_DM_00951]	Configuration of DTC "current FDC" as extended data record
[SWS_DM_00952]	Configuration of DTC "max. FDC since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. FDC current cycle" as extended data record
[SWS_DM_00954]	Configuration of DTC "occurrence counter" as extended data record
[SWS_DM_00955]	Configuration of DTC "aging counter up/down" as extended data record
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00957]	Configuration of DTC "aging counter down" as extended data record
[SWS_DM_00958]	Default value for DTC "aging counter up" if aging is not allowed
[SWS_DM_00959]	Default value for DTC "aging counter down" if aging is not allowed
[SWS_DM_00961]	Configuration of a DTCs significance as extended data record
[SWS_DM_00962]	Configuration of a DTCs Failed Operation Cycles as extended data record
[SWS_DM_00963]	Configuration of a DTCs failed operation Cycles Since First Failed as extended data record
[SWS_DM_00964]	Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record
[SWS_DM_00965]	Caching of monitor results
[SWS_DM_00966]	Reporting of DTCStatusAvailabilityMask
[SWS_DM_00967]	Support of UDS service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00970]	Behavior of failed data element retrieval
[SWS_DM_00971]	Definition of API class ara::diag::MetaInfo
[SWS_DM_00972]	Definition of API function ara::diag::MetaInfo::MetaInfo
[SWS_DM_00973]	Definition of API function ara::diag::MetaInfo::MetaInfo
[SWS_DM_00974]	Definition of API function ara::diag::MetaInfo::MetaInfo
[SWS_DM_00975]	Definition of API function ara::diag::MetaInfo::operator=
[SWS_DM_00976]	Definition of API function ara::diag::MetaInfo::operator=
[SWS_DM_00977]	Definition of API type ara::diag::MetaInfo::Context
[SWS_DM_00978]	Definition of API function ara::diag::MetaInfo::GetValue
[SWS_DM_00979]	Definition of API function ara::diag::MetaInfo::GetContext
[SWS_DM_00980]	Definition of API function ara::diag::MetaInfo::~MetaInfo
[SWS_DM_00981]	Conditions of status based reporting order

▽

△

Number	Heading
[SWS_DM_00982]	Reporting order direction
[SWS_DM_00983]	Processing of Custom Diagnostic Services
[SWS_DM_00984]	Return of cancellation status
[SWS_DM_00985]	Definition of API class ara::diag::DiagOfferException
[SWS_DM_00986]	Definition of API function ara::diag::DiagOfferException::DiagOfferException
[SWS_DM_00987]	Definition of API class ara::diag::DiagReportingException
[SWS_DM_00988]	Definition of API function ara::diag::DiagReportingException::DiagReporting Exception
[SWS_DM_00989]	Definition of API class ara::diag::DiagOfferErrorDomain
[SWS_DM_00990]	Definition of API type ara::diag::DiagOfferErrorDomain::Errc
[SWS_DM_00991]	Definition of API type ara::diag::DiagOfferErrorDomain::Exception
[SWS_DM_00992]	Definition of API function ara::diag::DiagOfferErrorDomain::DiagOfferError Domain
[SWS_DM_00993]	Definition of API function ara::diag::DiagOfferErrorDomain::Name
[SWS_DM_00994]	Definition of API function ara::diag::DiagOfferErrorDomain::Message
[SWS_DM_00995]	Definition of API function ara::diag::DiagOfferErrorDomain::ThrowAs Exception
[SWS_DM_00996]	Definition of API class ara::diag::DiagReportingErrorDomain
[SWS_DM_00997]	Definition of API type ara::diag::DiagReportingErrorDomain::Errc
[SWS_DM_00998]	Definition of API type ara::diag::DiagReportingErrorDomain::Exception
[SWS_DM_00999]	Definition of API function ara::diag::DiagReportingErrorDomain::Diag ReportingErrorDomain
[SWS_DM_01000]	Definition of API function ara::diag::DiagReportingErrorDomain::Name
[SWS_DM_01001]	Definition of API function ara::diag::DiagReportingErrorDomain::Message
[SWS_DM_01002]	Definition of API function ara::diag::DiagReportingErrorDomain::ThrowAs Exception
[SWS_DM_01003]	Definition of API function ara::diag::GetDiagOfferDomain
[SWS_DM_01004]	Definition of API function ara::diag::GetDiagReportingDomain
[SWS_DM_01005]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_01006]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_01007]	Definition of API type ara::diag::ResetRequestType
[SWS_DM_01009]	Definition of API class ara::diag::EcuResetRequest
[SWS_DM_01010]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01011]	Definition of API function ara::diag::EcuResetRequest::~EcuResetRequest
[SWS_DM_01012]	Definition of API function ara::diag::EcuResetRequest::EnableRapid Shutdown
[SWS_DM_01013]	Definition of API function ara::diag::EcuResetRequest::RequestReset
[SWS_DM_01014]	Definition of API function ara::diag::EcuResetRequest::ExecuteReset
[SWS_DM_01016]	Definition of API function ara::diag::EcuResetRequest::Offer
[SWS_DM_01017]	Definition of API function ara::diag::EcuResetRequest::StopOffer

▽

△

Number	Heading
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after `ara::diag::EcuResetRequest::RequestReset` called
[SWS_DM_01023]	Calling ExecuteReset() if positive response shall be sent before reset
[SWS_DM_01024]	Event Status processing
[SWS_DM_01025]	`Event status` bit transitions triggered by test results
[SWS_DM_01026]	Resetting the status of an `Event`
[SWS_DM_01027]	`Event status` bit transitions triggered by `operation cycle` restarting
[SWS_DM_01028]	`Event status` bit transitions triggered by ClearDiagnosticInformation `UDS service`
[SWS_DM_01029]	Notification about `Event status` changes
[SWS_DM_01030]	Observability of the `UDS DTC status byte`
[SWS_DM_01031]	Notification about UDS `DTC` status changes
[SWS_DM_01032]	Handling of 'WIR' bit without connected indicators
[SWS_DM_01033]	User controlled set of WIR-bit
[SWS_DM_01034]	User controlled reset of WIR-bit
[SWS_DM_01035]	User controlled WIR-bit handling and ControlDTCSetting
[SWS_DM_01037]	Behavior of not configured `DiagnosticEvent.confirmationThreshold`
[SWS_DM_01038]	Reading Diagnostic Data Identifier by `ara::diag::GenericDataIdentifier` interface
[SWS_DM_01039]	Writing Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_01040]	Realization of `UDS` service ReadDataByPeriodicIdentifier(0x2A)
[SWS_DM_01041]	Check requested number of periodic DataIdentifiers
[SWS_DM_01042]	Minimum length check for ReadDataByPeriodicIdentifier
[SWS_DM_01043]	Check supported periodic DataIdentifier
[SWS_DM_01044]	Check Transmission Mode
[SWS_DM_01045]	Check Scheduler Availability
[SWS_DM_01046]	Check supported DataIdentifier in active session
[SWS_DM_01047]	Check supported DataIdentifier on active security level
[SWS_DM_01048]	Check DataIdentifier for environmental conditions
[SWS_DM_01049]	Checks Dynamically Defined `DIDs` in ReadDataByPeriodicIdentifier
[SWS_DM_01050]	Periodic `DID` length check
[SWS_DM_01051]	DM behavior on transmission Mode stopSending without periodicData Identifier in the request
[SWS_DM_01052]	DM behavior on transmission Mode stopSending with supported periodic DataIdentifier in the request
[SWS_DM_01053]	DM behavior on transmission Mode stopSending with not supported periodic DataIdentifier in the request
[SWS_DM_01054]	Starting to transmit `PDIDs` after positive response

▽

△

Number	Heading
[SWS_DM_01055]	Reaction on ApplicationError
[SWS_DM_01056]	Optional condition checks for sending periodic DIDs
[SWS_DM_01057]	Optional stopping PDIDs after session change
[SWS_DM_01058]	Optional stopping PDIDs after security level change
[SWS_DM_01059]	No periodic DIDs in default session
[SWS_DM_01060]	Support of Scheduler type 1
[SWS_DM_01061]	Trigger all scheduled PDIDs per scheduler
[SWS_DM_01062]	Transmission of all PDIDs on the periodic connection
[SWS_DM_01063]	Transmission error behavior
[SWS_DM_01064]	Definition of API class apext::diag::uds_transport::UdsTransportProtocol PeriodicHandler
[SWS_DM_01065]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol PeriodicHandler::GetNumberOfPeriodicMessages
[SWS_DM_01066]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol PeriodicHandler::GetMaxPayloadLength
[SWS_DM_01067]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol PeriodicHandler::PeriodicTransmit
[SWS_DM_01068]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::GetPeriodicHandler
[SWS_DM_01069]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::PeriodicTransmitConfirmation
[SWS_DM_01070]	Support of UDS service 0x2C in Adaptive AUTOSAR DM
[SWS_DM_01071]	No persistency of defined DDDIDs
[SWS_DM_01072]	Persistency of defined DDDIDs
[SWS_DM_01073]	DM behavior for subfunction 'defineByIdentifier'
[SWS_DM_01074]	Only static DIDs as sourceDataIdentifier
[SWS_DM_01075]	Maximum number of sourceDataIdentifiers in the request
[SWS_DM_01076]	Clearing all configured DDDIDs
[SWS_DM_01077]	Clearing individual configured DDDIDs
[SWS_DM_01078]	Clear DDDIDs on session change
[SWS_DM_01079]	Session check for DDDID
[SWS_DM_01080]	Security level check for DDDID
[SWS_DM_01081]	Session check for sourceDataIdentifier
[SWS_DM_01082]	Security level check for sourceDataIdentifier
[SWS_DM_01083]	Use of configured DID ports to get DDDID data
[SWS_DM_01085]	Triggering for snapshot record storage (configured, without update)
[SWS_DM_01086]	Triggering for snapshot record storage (configured, with update)
[SWS_DM_01087]	Snapshot record layout
[SWS_DM_01088]	Definition of API function ara::diag::Monitor::Offer
[SWS_DM_01089]	Definition of API function ara::diag::Monitor::StopOffer

▽

△

Number	Heading
[SWS_DM_01090]	Calling ExecuteReset() if positive response shall be sent after reset
[SWS_DM_01092]	EnableRapidPowerShutdown Positive Response
[SWS_DM_01093]	Caching of conditions
[SWS_DM_01094]	Monitor initialization for enable condition re-enabling reason and Control DTCSetting set to On
[SWS_DM_01095]	Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off
[SWS_DM_01096]	UDS service TransferData (0x36) upload processing
[SWS_DM_01097]	UDS service RequestTransferExit (0x37) upload processing
[SWS_DM_01098]	Starting ResponseOnEvent in single and multiple client scenarios
[SWS_DM_01099]	Debouncing parameters from DEXT
[SWS_DM_01101]	Debouncing parameters from Monitor Constructor
[SWS_DM_01102]	Definition of API function ara::diag::OperationCycle::RestartOperationCycle
[SWS_DM_01103]	Caching of RestartOperationCycle
[SWS_DM_01104]	Operation Cycle restart
[SWS_DM_01105]	Restart OperationCycle during the processing of previous call
[SWS_DM_01106]	Applicability of Event Combination
[SWS_DM_01107]	DTC Status Byte calculation
[SWS_DM_01108]	Clear all DTCs event with event combination
[SWS_DM_01109]	UDS DTC status update for combined DTCs
[SWS_DM_01110]	Callbacks for combined UDS DTC status change
[SWS_DM_01111]	Fault detection counter for combined events
[SWS_DM_01112]	Event memory entry for events with the combination on storage
[SWS_DM_01113]	Aging counter for combined events
[SWS_DM_01114]	Data storage for event combination on retrieval
[SWS_DM_01115]	Data reporting for event combination on retrieval
[SWS_DM_01116]	Reporting order of snapshot and extended data records
[SWS_DM_01117]	Support of eventWindowTime values for ResponseOnEvent
[SWS_DM_01118]	Support of DEXT parameter DiagnosticResponseOnEventClass.responseOn EventSchedulerRate
[SWS_DM_01119]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNum ChangeOfDataIdentfierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNum ComparisionOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.max SupportedDIDLength
[SWS_DM_01122]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumber OfStoredDTCStatusChangedEvents
[SWS_DM_01123]	Definition of API class ara::diag::Authentication
[SWS_DM_01124]	Definition of API function ara::diag::Authentication::Authentication

▽

△

Number	Heading
[SWS_DM_01125]	Definition of API function ara::diag::Authentication::~Authentication
[SWS_DM_01126]	Definition of API function ara::diag::Authentication::VerifyCertificate Unidirectional
[SWS_DM_01127]	Definition of API function ara::diag::Authentication::VerifyCertificate Bidirectional
[SWS_DM_01128]	Definition of API function ara::diag::Authentication::VerifyOwnership
[SWS_DM_01130]	Definition of API function ara::diag::Authentication::Offer
[SWS_DM_01131]	Definition of API function ara::diag::Authentication::StopOffer
[SWS_DM_01132]	Definition of API class ara::diag::ClientAuthentication
[SWS_DM_01133]	Definition of API enum ara::diag::ClientAuthentication::DiagnosticAuthState
[SWS_DM_01134]	Definition of API type ara::diag::ClientAuthentication::DiagnosticAuthRole
[SWS_DM_01136]	Definition of API function ara::diag::ClientAuthentication::~Client Authentication
[SWS_DM_01137]	Definition of API function ara::diag::ClientAuthentication::Client Authentication
[SWS_DM_01138]	Definition of API function ara::diag::ClientAuthentication::operator=
[SWS_DM_01139]	Definition of API function ara::diag::ClientAuthentication::Client Authentication
[SWS_DM_01140]	Definition of API function ara::diag::ClientAuthentication::operator=
[SWS_DM_01141]	Definition of API function ara::diag::ClientAuthentication::OverrideDefault Roles
[SWS_DM_01142]	Definition of API function ara::diag::ClientAuthentication::Authenticate
[SWS_DM_01143]	Definition of API function ara::diag::ClientAuthentication::GetState
[SWS_DM_01144]	Definition of API function ara::diag::ClientAuthentication::SetNotifier
[SWS_DM_01145]	Definition of API class ara::diag::ClientAuthenticationHandle
[SWS_DM_01146]	Definition of API function ara::diag::ClientAuthenticationHandle::Client AuthenticationHandle
[SWS_DM_01147]	Definition of API function ara::diag::ClientAuthenticationHandle::~Client AuthenticationHandle
[SWS_DM_01148]	Definition of API function ara::diag::ClientAuthenticationHandle::Client AuthenticationHandle
[SWS_DM_01149]	Definition of API function ara::diag::ClientAuthenticationHandle::operator=
[SWS_DM_01150]	Definition of API function ara::diag::ClientAuthenticationHandle::Client AuthenticationHandle
[SWS_DM_01151]	Definition of API function ara::diag::ClientAuthenticationHandle::operator=
[SWS_DM_01152]	Definition of API function ara::diag::ClientAuthenticationHandle::Append
[SWS_DM_01153]	Definition of API function ara::diag::ClientAuthenticationHandle::Set
[SWS_DM_01154]	Definition of API function ara::diag::ClientAuthenticationHandle::Revoke
[SWS_DM_01155]	Definition of API function ara::diag::ClientAuthenticationHandle::Refresh
[SWS_DM_01156]	Definition of API class ara::diag::DiagnosticServiceDynamicAccessList
[SWS_DM_01157]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::DiagnosticServiceDynamicAccessList

▽

△

Number	Heading
[SWS_DM_01158]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::~DiagnosticServiceDynamicAccessList
[SWS_DM_01159]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::DiagnosticServiceDynamicAccessList
[SWS_DM_01160]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::DiagnosticServiceDynamicAccessList
[SWS_DM_01161]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::operator=
[SWS_DM_01162]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::operator=
[SWS_DM_01163]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::Reserve
[SWS_DM_01164]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::MakeServiceBuilder
[SWS_DM_01165]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::MakeServiceBuilder
[SWS_DM_01166]	Definition of API class ara::diag::DynamicAccessListDiagServiceBuilder
[SWS_DM_01167]	Definition of API type ara::diag::DynamicAccessListDiagServiceBuilder::Byte
[SWS_DM_01168]	Definition of API type ara::diag::DynamicAccessListDiagServiceBuilder::Byte String
[SWS_DM_01169]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::DynamicAccessListDiagServiceBuilder
[SWS_DM_01170]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::DynamicAccessListDiagServiceBuilder
[SWS_DM_01171]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::DynamicAccessListDiagServiceBuilder
[SWS_DM_01172]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::operator=
[SWS_DM_01173]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::operator=
[SWS_DM_01174]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::~DynamicAccessListDiagServiceBuilder
[SWS_DM_01175]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::Add
[SWS_DM_01176]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::Add
[SWS_DM_01177]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::Add
[SWS_DM_01178]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::Any
[SWS_DM_01179]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::EndsWith
[SWS_DM_01180]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::EndsWith

▽

△

Number	Heading
[SWS_DM_01181]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::Build
[SWS_DM_01182]	Definition of API class ara::diag::DynamicAccessListDiagService Builder::ByteRange
[SWS_DM_01183]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::ByteRange
[SWS_DM_01184]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::ByteRange
[SWS_DM_01185]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::ByteRange
[SWS_DM_01186]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::operator=
[SWS_DM_01187]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::operator=
[SWS_DM_01188]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::~ByteRange
[SWS_DM_01189]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::GetMin
[SWS_DM_01190]	Definition of API function ara::diag::DynamicAccessListDiagService Builder::ByteRange::GetMax
[SWS_DM_01191]	Definition of API class ara::diag::ExternalAuthentication
[SWS_DM_01192]	Definition of API type ara::diag::ExternalAuthentication::Address
[SWS_DM_01193]	Definition of API function ara::diag::ExternalAuthentication::External Authentication
[SWS_DM_01194]	Definition of API function ara::diag::ExternalAuthentication::External Authentication
[SWS_DM_01195]	Definition of API function ara::diag::ExternalAuthentication::operator=
[SWS_DM_01196]	Definition of API function ara::diag::ExternalAuthentication::External Authentication
[SWS_DM_01197]	Definition of API function ara::diag::ExternalAuthentication::operator=
[SWS_DM_01198]	Definition of API function ara::diag::ExternalAuthentication::~External Authentication
[SWS_DM_01199]	Definition of API function ara::diag::ExternalAuthentication::Get
[SWS_DM_01200]	Definition of API function ara::diag::ExternalAuthentication::Get
[SWS_DM_01201]	Definition of API function ara::diag::ExternalAuthentication::GetAll
[SWS_DM_01202]	Get ClientAuthentication Instance
[SWS_DM_01203]	GetAll ClientAuthentication Instance
[SWS_DM_01204]	Default Authentication Role
[SWS_DM_01205]	Default Authentication State
[SWS_DM_01206]	Set AuthenticationRole
[SWS_DM_01207]	Get Authentication State
[SWS_DM_01208]	Authentication State Change Notifier

▽

△

Number	Heading
[SWS_DM_01209]	Temporarily change Default Roles
[SWS_DM_01210]	DeAuthenticate due to client inactivity
[SWS_DM_01211]	Transition to DeAuthenticated state on S3server timeout
[SWS_DM_01212]	Transition from Authenticated to DeAuthenticated State
[SWS_DM_01213]	Set DynamicAccessList
[SWS_DM_01214]	Default DynamicAccessList
[SWS_DM_01215]	Extend the DynamicAccessList
[SWS_DM_01216]	Revoke an authentication
[SWS_DM_01217]	Refresh timeouts
[SWS_DM_01218]	Building a new DynamicAccessList
[SWS_DM_01219]	Adding patterns to a DynamicAccessList
[SWS_DM_01220]	Adding wildcards to a DynamicAccessList
[SWS_DM_01221]	End patterns of a DynamicAccessList
[SWS_DM_01222]	Finalize a DynamicAccessList
[SWS_DM_01223]	Diagnostic service role verification
[SWS_DM_01224]	Diagnostic service dynamic access-rights verification
[SWS_DM_01225]	Response behavior of services without access rights
[SWS_DM_01226]	Support of UDS service authentication
[SWS_DM_01227]	Configuration of authentication types
[SWS_DM_01228]	Mandatory sub functions
[SWS_DM_01229]	Support for authentication per Diagnostic Client
[SWS_DM_01230]	Processing the verifyCertificateUnidirectional request
[SWS_DM_01231]	Handling Negative return values of `ara::diag::Authentication::VerifyCertificateUnidirectional` method
[SWS_DM_01233]	Successful verification of verifyCertificateUnidirectional
[SWS_DM_01235]	Processing the verifyCertificateBidirectional request
[SWS_DM_01236]	Handling Negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method
[SWS_DM_01238]	Successful verification of verifyCertificateBidirectional
[SWS_DM_01240]	Processing the proofOfOwnership request
[SWS_DM_01241]	Handling Negative return values of `ara::diag::Authentication::VerifyOwnership` method
[SWS_DM_01243]	Successful verification of Client proofOfOwnership
[SWS_DM_01244]	Processing the deAuthenticate request
[SWS_DM_01245]	Successful completion of deAuthenticate
[SWS_DM_01246]	Processing the authenticationConfiguration request
[SWS_DM_01247]	Validation of the transmitCertificate certificateEvaluationId
[SWS_DM_01248]	Processing the transmitCertificate request
[SWS_DM_01249]	Handling Negative return values of `ara::diag::TransmitCertificate::Process` method

▽

△

Number	Heading
[SWS_DM_01251]	Successful verification of transmitCertificate
[SWS_DM_01252]	Support of manufacturer service validations
[SWS_DM_01253]	Support of supplier service validations
[SWS_DM_01254]	Continue service processing after validation
[SWS_DM_01255]	`NRC` after failed service validation
[SWS_DM_01256]	Support of UDS service ReadDTCInformation, Subfunction 0x03
[SWS_DM_01257]	ResourceTemporarilyNotAvailable NRC handling
[SWS_DM_01258]	Response handling
[SWS_DM_01259]	Validation of Security Level Locked in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`
[SWS_DM_01260]	Validation of Invalid Security Level in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`
[SWS_DM_01261]	Validation of Invalid Session Level in `ara::diag::Conversation::GetDiagnosticSessionShortName`
[SWS_DM_01262]	No storage of RoE events
[SWS_DM_01263]	Storage of RoE events
[SWS_DM_01264]	`DiagnosticAging.threshold` reached
[SWS_DM_01265]	`Aging` requires tested cycles only
[SWS_DM_01266]	Warning Indicator Request Activation
[SWS_DM_01267]	Reporting `kFdcThresholdReached` for monitor internal debouncing
[SWS_DM_01268]	Value of FaultDetectionCounter in case of monitor internal debouncing
[SWS_DM_01269]	Requesting `DTC` number for events without `DTC`
[SWS_DM_01270]	UDS Service RoutineControl (0x31) startRoutine processing with typed interface
[SWS_DM_01271]	UDS Service RoutineControl (0x31) stopRoutine processing with typed interface
[SWS_DM_01272]	UDS Service RoutineControl (0x31) requestRoutineResults processing with typed interface
[SWS_DM_01276]	Triggering for `snapshot record` storage (calculated, maxNumberFreeze FrameRecords = 1)
[SWS_DM_01277]	Triggering for `snapshot record` storage (calculated, maxNumberFreeze FrameRecords > 1)
[SWS_DM_01278]	Definition of API variable ara::diag::kDefaultSession
[SWS_DM_01279]	Definition of API variable ara::diag::kProgrammingSession
[SWS_DM_01280]	Definition of API variable ara::diag::kExtendedDiagnosticSession
[SWS_DM_01281]	Definition of API variable ara::diag::kSafetySystemDiagnosticSession
[SWS_DM_01282]	Definition of API variable ara::diag::kLocked
[SWS_DM_01283]	Definition of API variable ara::diag::kSoftReset
[SWS_DM_01284]	Definition of API variable ara::diag::kHardReset
[SWS_DM_01285]	Definition of API variable ara::diag::kKeyOffOnReset
[SWS_DM_01286]	Definition of API variable ara::diag::kCustomReset

▽

△

Number	Heading
[SWS_DM_01309]	Unblock requests after `ara::diag::EcuResetRequest::ExecuteReset` completed
[SWS_DM_01310]	Realization of UDS service RequestFileTransfer (0x38)
[SWS_DM_01311]	Realization of modeOfOperation AddFile (0x01)
[SWS_DM_01312]	Realization of modeOfOperation DeleteFile (0x02)
[SWS_DM_01313]	Realization of modeOfOperation ReplaceFile (0x03)
[SWS_DM_01314]	Realization of modeOfOperation ReadFile (0x04)
[SWS_DM_01315]	Realization of modeOfOperation ReadDir (0x05)
[SWS_DM_01316]	Realization of modeOfOperation ResumeFile (0x06)
[SWS_DM_01317]	Realization of TransferData (0x36) in context of RequestFileTransfer
[SWS_DM_01318]	Realization of RequestTransferExit (0x37) in context of RequestFileTransfer
[SWS_DM_01319]	Consecutive registration of notifier with ReleaseHandler::SetNotifier()
[SWS_DM_01320]	Definition of API class ara::diag::FileTransferService
[SWS_DM_01321]	Definition of API class ara::diag::FileTransferService::FileSizes
[SWS_DM_01322]	Definition of API variable ara::diag::FileTransferService::FileSizes::uncompressed_size
[SWS_DM_01323]	Definition of API variable ara::diag::FileTransferService::FileSizes::compressed_size
[SWS_DM_01324]	Definition of API enum ara::diag::FileTransferService::WriteFileMode
[SWS_DM_01325]	Definition of API function ara::diag::FileTransferService::FileTransferService
[SWS_DM_01326]	Definition of API function ara::diag::FileTransferService::~FileTransferService
[SWS_DM_01327]	Definition of API function ara::diag::FileTransferService::FileTransferService
[SWS_DM_01328]	Definition of API function ara::diag::FileTransferService::FileTransferService
[SWS_DM_01329]	Definition of API function ara::diag::FileTransferService::operator=
[SWS_DM_01330]	Definition of API function ara::diag::FileTransferService::operator=
[SWS_DM_01331]	Definition of API function ara::diag::FileTransferService::RequestReadFile
[SWS_DM_01332]	Definition of API function ara::diag::FileTransferService::RequestReadDirectory
[SWS_DM_01333]	Definition of API function ara::diag::FileTransferService::RequestWriteFile
[SWS_DM_01334]	Definition of API function ara::diag::FileTransferService::RequestResumeWriteFile
[SWS_DM_01335]	Definition of API function ara::diag::FileTransferService::DeleteFile
[SWS_DM_01336]	Definition of API function ara::diag::FileTransferService::Offer
[SWS_DM_01337]	Definition of API function ara::diag::FileTransferService::StopOffer
[SWS_DM_01339]	Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession
[SWS_DM_01340]	Definition of API class ara::diag::ReleaseHandler
[SWS_DM_01342]	Definition of API variable ara::diag::MetaInfo::kDiagnosticCommunication
[SWS_DM_01343]	Definition of API variable ara::diag::MetaInfo::kFaultMemory

▽

△

Number	Heading
[SWS_DM_01344]	Definition of API variable ara::diag::MetaInfo::kDoIP
[SWS_DM_01345]	Lifetime of MetaInfo
[SWS_DM_01346]	Handling negative return values of ara::diag::EcuResetRequest::Execute Reset
[SWS_DM_01347]	Handling unspecified negative return values of ara::diag::EcuReset Request::ExecuteReset
[SWS_DM_01348]	Consecutive registration of notifier with CancellationHandler::SetNotifier()
[SWS_DM_01349]	Consecutive registration of notifier with SetEventStatusChangedNotifier()
[SWS_DM_01350]	Consecutive registration of notifier with SetDTCStatusChangedNotifier()
[SWS_DM_01351]	Consecutive registration of notifier with SetSnapshotRecordUpdated Notifier()
[SWS_DM_01352]	Consecutive registration of notifier with SetNumberOfStoredEntriesNotifier()
[SWS_DM_01353]	Consecutive registration of notifier with SetControlDtcStatusNotifier()
[SWS_DM_01354]	Consecutive registration of notifier with SetEventMemoryOverflowNotifier()
[SWS_DM_01355]	Consecutive registration of notifier with SetActivityNotifier()
[SWS_DM_01356]	Consecutive registration of notifier with SetDiagnosticSessionNotifier()
[SWS_DM_01357]	Consecutive registration of notifier with SetSecurityLevelNotifier()
[SWS_DM_01358]	Consecutive registration of notifier with OperationCycle::SetNotifier()
[SWS_DM_01359]	Consecutive registration of notifier with Indicator::SetNotifier()
[SWS_DM_01360]	Consecutive registration of notifier with ClientAuthentication::SetNotifier()
[SWS_DM_01361]	Providing EID in DoIP protocol messages using Application Interface
[SWS_DM_01362]	Definition of API class ara::diag::DoIPEntityIdentification
[SWS_DM_01363]	Definition of API class ara::diag::DoIPEntityIdentification::EntityId
[SWS_DM_01364]	Definition of API function ara::diag::DoIPEntityIdentification::DoIPEntity Identification
[SWS_DM_01365]	Definition of API function ara::diag::DoIPEntityIdentification::~DoIPEntity Identification
[SWS_DM_01366]	Definition of API function ara::diag::DoIPEntityIdentification::GetEntityId
[SWS_DM_01367]	Definition of API function ara::diag::DoIPEntityIdentification::Offer
[SWS_DM_01368]	Definition of API function ara::diag::DoIPEntityIdentification::StopOffer
[SWS_DM_01369]	DM as SOVD Server
[SWS_DM_01370]	DNS-Based Service Discovery and Multicast DNS for SOVD
[SWS_DM_01371]	Secure Communication for SOVD using TLS
[SWS_DM_01372]	Representation of DM by SOVD component
[SWS_DM_01373]	Representation of Diagnostic Server instance by SOVD subcomponents
[SWS_DM_01374]	Dispatching of SOVD requests/responses
[SWS_DM_01375]	Behavior on locked SOVD
[SWS_DM_01376]	Response behavior of SOVD services without access rights
[SWS_DM_01379]	SOVD mode dtcsetting set value

▽

△

Number	Heading
[SWS_DM_01380]	SOVD mode dtcsetting get value
[SWS_DM_01381]	SOVD mode dtcsetting value schema
[SWS_DM_01382]	SOVD mode dtcsetting name
[SWS_DM_01383]	SOVD mode dtcsetting
[SWS_DM_01384]	SOVD mode commctrl set value
[SWS_DM_01385]	SOVD mode commctrl get value
[SWS_DM_01386]	SOVD mode commctrl value schema
[SWS_DM_01387]	SOVD mode commctrl name
[SWS_DM_01388]	SOVD mode commctrl
[SWS_DM_01389]	SOVD method Retrieve List of All Supported Modes of an Entity
[SWS_DM_01390]	SOVD operations request and response parameters
[SWS_DM_01391]	SOVD method Stop the Execution of an Operation proximity_response
[SWS_DM_01392]	SOVD method Stop the Execution of an Operation
[SWS_DM_01393]	SOVD method Get the Status of an Operation Execution
[SWS_DM_01394]	SOVD method Get Executions of an Operation
[SWS_DM_01395]	SOVD method Start Execution of an Operation proximity_response
[SWS_DM_01396]	SOVD method Start Execution of an Operation
[SWS_DM_01397]	SOVD operation mode
[SWS_DM_01398]	SOVD operation proximity_challenge
[SWS_DM_01399]	SOVD operation attribute asynchronous_execution
[SWS_DM_01401]	SOVD operation attribute name
[SWS_DM_01402]	SOVD operation attribute id
[SWS_DM_01403]	SOVD method Get Details of a Single Operation
[SWS_DM_01404]	SOVD method Retrieve List of All Available Operations from an Entity
[SWS_DM_01405]	SOVD operations
[SWS_DM_01406]	Support of SOVD method Delete Single Fault of an Entity
[SWS_DM_01409]	Support of SOVD method Delete All Faults of an Entity
[SWS_DM_01411]	SOVD fault environment_data query
[SWS_DM_01412]	SOVD fault environment_data
[SWS_DM_01414]	SOVD fault attribute status
[SWS_DM_01415]	SOVD fault general attributes
[SWS_DM_01416]	Support of SOVD method Read Details for a Fault
[SWS_DM_01417]	Support of SOVD method Read Faults from an Entity
[SWS_DM_01418]	SOVD faults
[SWS_DM_01419]	SOVD method Read Multiple Data Values at Once from an Entity Using a Data List

▽

△

Number	Heading
[SWS_DM_01420]	SOVD data-list
[SWS_DM_01421]	SOVD method Write Configuration as Parameters data processing
[SWS_DM_01422]	SOVD method Write Configuration as Parameters
[SWS_DM_01423]	SOVD method Read Configuration as Parameters data query
[SWS_DM_01424]	SOVD method Read Configuration as Parameters
[SWS_DM_01425]	SOVD configuration attribute version and content_type
[SWS_DM_01426]	SOVD configuration attribute type
[SWS_DM_01427]	SOVD configuration attribute name
[SWS_DM_01428]	SOVD configuration attribute id
[SWS_DM_01429]	SOVD method Retrieve List of All Configurations Provided by the Entity
[SWS_DM_01430]	SOVD method Write a Data Value to an Entity data processing
[SWS_DM_01431]	SOVD method Write a Data Value to an Entity
[SWS_DM_01432]	SOVD method Read Single Data Value from an Entity data query
[SWS_DM_01433]	SOVD method Read Single Data Value from an Entity
[SWS_DM_01434]	SOVD data attribute data of internal structure
[SWS_DM_01435]	SOVD group id
[SWS_DM_01436]	SOVD group attribute category
[SWS_DM_01437]	SOVD group category uniqueness
[SWS_DM_01440]	SOVD data attribute name
[SWS_DM_01441]	SOVD data attribute id
[SWS_DM_01442]	SOVD method Retrieve List of All Data Provided by the Entity
[SWS_DM_01443]	SOVD method for locking
[SWS_DM_01444]	Data-groups type content
[SWS_DM_01445]	Data-groups type
[SWS_DM_01446]	SOVD method Retrieve Groups Supported by a Data Resource Collection
[SWS_DM_01447]	Data categories type
[SWS_DM_01448]	Standard resource Data categories
[SWS_DM_01449]	SOVD method Retrieve Categories Supported by a Data Resource Collection
[SWS_DM_01450]	SOVDInfo type content
[SWS_DM_01451]	SOVDInfo type
[SWS_DM_01452]	Mismatching versions
[SWS_DM_01453]	path to version-info
[SWS_DM_01454]	Query to the SOVD API version
[SWS_DM_01455]	SOVD method for SOVD API Versioning
[SWS_DM_01456]	SOVD method Query an Online Capability Description
[SWS_DM_01457]	SOVD data representation of arrays

▽

△

Number	Heading
[SWS_DM_01458]	SOVD data representation of units
[SWS_DM_01459]	SOVD data representation of textual Strings
[SWS_DM_01460]	SOVD data representation of baseTypeEncoding IEEE754
[SWS_DM_01461]	SOVD data representation of BITFIELD_TEXTTABLE
[SWS_DM_01462]	SOVD data representation of TAB_NOINTP
[SWS_DM_01463]	SOVD data representation of atomic scaled numeric data with texttable
[SWS_DM_01464]	SOVD data representation of TEXTTABLE
[SWS_DM_01465]	SOVD data representation of atomic numeric data
[SWS_DM_01466]	Environmental Condition Check for SOVD evaluated to FALSE
[SWS_DM_01467]	Environmental Condition Check for SOVD evaluated to TRUE
[SWS_DM_01468]	Check of Environmental Conditions before executing SOVD methods
[SWS_DM_01469]	Validity period of authenticated roles
[SWS_DM_01470]	Authorization validation
[SWS_DM_01471]	Redirection to token endpoint
[SWS_DM_01472]	Redirection to authorization endpoint
[SWS_DM_01473]	DM shall lock SOVD entity after time expiration
[SWS_DM_01474]	DM shall allow access only to unlocked SOVD entities
[SWS_DM_01475]	DM shall allow only one lock per SOVD entity
[SWS_DM_01476]	Parallel SOVD client handling
[SWS_DM_01477]	SOVD lock in UDS extended session
[SWS_DM_01478]	Definition of API function ara::diag::SovdProximityChallenge::Offer
[SWS_DM_01479]	Definition of API function ara::diag::SovdProximityChallenge::~Sovd ProximityChallenge
[SWS_DM_01480]	Definition of API function ara::diag::SovdProximityChallenge::SovdProximity Challenge
[SWS_DM_01481]	Definition of API class ara::diag::SovdProximityChallenge
[SWS_DM_01482]	Definition of API class ara::diag::SovdProximityChallengeType
[SWS_DM_01483]	Definition of API function ara::diag::SovdAuthorization::ValidateAuthorization
[SWS_DM_01484]	Definition of API function ara::diag::SovdAuthorization::GetTokenUrl
[SWS_DM_01485]	Definition of API function ara::diag::SovdAuthorization::GetAuthorizationUrl
[SWS_DM_01486]	Definition of API function ara::diag::SovdAuthorization::StopOffer
[SWS_DM_01487]	Definition of API function ara::diag::SovdAuthorization::Offer
[SWS_DM_01488]	Definition of API function ara::diag::SovdAuthorization::~SovdAuthorization
[SWS_DM_01489]	Definition of API function ara::diag::SovdAuthorization::SovdAuthorization
[SWS_DM_01490]	Definition of API class ara::diag::SovdAuthorization
[SWS_DM_01491]	Definition of API variable ara::diag::SovdProximityChallengeType::valid_until
[SWS_DM_01492]	Definition of API variable ara::diag::SovdProximityChallengeType::challenge
[SWS_DM_01493]	Definition of API function ara::diag::SovdProximityChallenge::Validate Response

▽

△

Number	Heading
[SWS_DM_01494]	Definition of API function ara::diag::SovdProximityChallenge::GetChallenge
[SWS_DM_01495]	Definition of API function ara::diag::SovdProximityChallenge::StopOffer
[SWS_DM_01496]	Definition of API variable ara::diag::DoIPEntityIdentification::EntityId::entity Identification
[SWS_DM_01497]	Definition of API class ara::diag::DataTransferReadSharedDataHandler
[SWS_DM_01498]	Definition of API function ara::diag::DataTransferReadSharedData Handler::DataTransferReadSharedDataHandler
[SWS_DM_01499]	Definition of API function ara::diag::DataTransferReadSharedData Handler::~DataTransferReadSharedDataHandler
[SWS_DM_01500]	Definition of API function ara::diag::DataTransferReadSharedData Handler::DataTransferReadSharedDataHandler
[SWS_DM_01501]	Definition of API function ara::diag::DataTransferReadSharedData Handler::DataTransferReadSharedDataHandler
[SWS_DM_01502]	Definition of API function ara::diag::DataTransferReadSharedData Handler::operator=
[SWS_DM_01503]	Definition of API function ara::diag::DataTransferReadSharedData Handler::operator=
[SWS_DM_01504]	Definition of API function ara::diag::DataTransferReadSharedData Handler::Read
[SWS_DM_01505]	Definition of API function ara::diag::DataTransferReadSharedData Handler::ExitRead
[SWS_DM_01506]	Definition of API class ara::diag::DataTransferReadByPullHandler
[SWS_DM_01507]	Definition of API function ara::diag::DataTransferReadByPullHandler::Data TransferReadByPullHandler
[SWS_DM_01508]	Definition of API function ara::diag::DataTransferReadByPullHandler::Data TransferReadByPullHandler
[SWS_DM_01509]	Definition of API function ara::diag::DataTransferReadByPullHandler::Data TransferReadByPullHandler
[SWS_DM_01510]	Definition of API function ara::diag::DataTransferReadByPullHandler::~Data TransferReadByPullHandler
[SWS_DM_01511]	Definition of API function ara::diag::DataTransferReadByPull Handler::operator=
[SWS_DM_01512]	Definition of API function ara::diag::DataTransferReadByPull Handler::operator=
[SWS_DM_01513]	Definition of API function ara::diag::DataTransferReadByPullHandler::Read
[SWS_DM_01514]	Definition of API function ara::diag::DataTransferReadByPullHandler::Exit Read
[SWS_DM_01515]	Definition of API class ara::diag::DataTransferReadByPushHandler
[SWS_DM_01516]	Definition of API function ara::diag::DataTransferReadByPushHandler::Data TransferReadByPushHandler
[SWS_DM_01517]	Definition of API function ara::diag::DataTransferReadByPushHandler::Data TransferReadByPushHandler
[SWS_DM_01518]	Definition of API function ara::diag::DataTransferReadByPushHandler::Data TransferReadByPushHandler

▽

△

Number	Heading
[SWS_DM_01519]	Definition of API function ara::diag::DataTransferReadByPush Handler::~DataTransferReadByPushHandler
[SWS_DM_01520]	Definition of API function ara::diag::DataTransferReadByPush Handler::operator=
[SWS_DM_01521]	Definition of API function ara::diag::DataTransferReadByPush Handler::operator=
[SWS_DM_01522]	Definition of API function ara::diag::DataTransferReadByPushHandler::Read
[SWS_DM_01523]	Definition of API function ara::diag::DataTransferReadByPushHandler::Exit Read
[SWS_DM_01524]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::~UdsTransportProtocolMgr
[SWS_DM_01525]	`ara::diag::DoIPPowerMode` not yet offered when client requests DoIP PowerMode
[SWS_DM_01526]	`ara::diag::DoIPActivationLine` not yet offered on activation line controlled network interfaces
[SWS_DM_01527]	`ara::diag::DoIPGroupIdentification` not yet offered when DM needs to retrieve GID
[SWS_DM_01528]	`ara::diag::DoIPEntityIdentification` not yet offered when DM needs to retrieve `EID` from Application
[SWS_DM_01529]	Behavior on failed ara::diag instantiation
[SWS_DM_01530]	Definition of API function ara::diag::ReleaseHandler::ReleaseHandler
[SWS_DM_01531]	Definition of API function ara::diag::ReleaseHandler::~ReleaseHandler
[SWS_DM_01532]	Definition of API function ara::diag::ReleaseHandler::ReleaseHandler
[SWS_DM_01533]	Definition of API function ara::diag::ReleaseHandler::operator=
[SWS_DM_01534]	Definition of API function ara::diag::ReleaseHandler::ReleaseHandler
[SWS_DM_01535]	Definition of API function ara::diag::ReleaseHandler::operator=
[SWS_DM_01536]	Definition of API function ara::diag::ReleaseHandler::MayRelease
[SWS_DM_01537]	Definition of API function ara::diag::ReleaseHandler::SetNotifier
[SWS_DM_01538]	Definition of API enum ara::diag::DataTransferExitType
[SWS_DM_01539]	Definition of API class ara::diag::DataTransferWriteHandler
[SWS_DM_01540]	Definition of API function ara::diag::DataTransferWriteHandler::DataTransfer WriteHandler
[SWS_DM_01541]	Definition of API function ara::diag::DataTransferWriteHandler::~Data TransferWriteHandler
[SWS_DM_01542]	Definition of API function ara::diag::DataTransferWriteHandler::DataTransfer WriteHandler
[SWS_DM_01543]	Definition of API function ara::diag::DataTransferWriteHandler::operator=
[SWS_DM_01544]	Definition of API function ara::diag::DataTransferWriteHandler::DataTransfer WriteHandler
[SWS_DM_01545]	Definition of API function ara::diag::DataTransferWriteHandler::operator=
[SWS_DM_01546]	Definition of API function ara::diag::DataTransferWriteHandler::Write
[SWS_DM_01547]	Definition of API function ara::diag::DataTransferWriteHandler::ExitWrite

▽

△

Number	Heading
[SWS_DM_01548]	Definition of API class ara::diag::DataTransferReadSession
[SWS_DM_01549]	Definition of API function ara::diag::DataTransferReadSession::DataTransfer ReadSession
[SWS_DM_01550]	Definition of API function ara::diag::DataTransferReadSession::~Data TransferReadSession
[SWS_DM_01551]	Definition of API function ara::diag::DataTransferReadSession::DataTransfer ReadSession
[SWS_DM_01552]	Definition of API function ara::diag::DataTransferReadSession::operator=
[SWS_DM_01553]	Definition of API function ara::diag::DataTransferReadSession::DataTransfer ReadSession
[SWS_DM_01554]	Definition of API function ara::diag::DataTransferReadSession::operator=
[SWS_DM_01555]	Definition of API class ara::diag::DataTransferWriteSession
[SWS_DM_01556]	Definition of API function ara::diag::DataTransferWriteSession::~Data TransferWriteSession
[SWS_DM_01557]	Definition of API function ara::diag::DataTransferWriteSession::DataTransfer WriteSession
[SWS_DM_01558]	Definition of API function ara::diag::DataTransferWriteSession::operator=
[SWS_DM_01559]	Definition of API function ara::diag::DataTransferWriteSession::DataTransfer WriteSession
[SWS_DM_01560]	Definition of API function ara::diag::DataTransferWriteSession::operator=
[SWS_DM_01561]	Response Body for SOVD fault environment_data table
[SWS_DM_01562]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::controlType
[SWS_DM_01563]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::communicationType
[SWS_DM_01564]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequest ParamsType::nodeIdentificationNumber
[SWS_DM_01565]	Supported internal DiagnosticDataElements
[SWS_DM_01566]	Subfunctions of 0x19 / ReadDTCInformation with chronolicical reporting order
[SWS_DM_01567]	Response Body for SOVD fault attributes
[SWS_DM_01568]	supportedDoIpMessageTypes
[SWS_DM_01569]	Configurable reset of the pendingDTC bit in case of displacement
[SWS_DM_01570]	Lifetime of overridden default roles
[SWS_DM_01571]	Recovery of persisted data
[SWS_DM_01572]	Graceful shutdown
[SWS_DM_01573]	Stop all running Transport Protocol Handlers
[SWS_DM_01574]	Write data to be persisted
[SWS_DM_01576]	Behavior of not configured DiagnosticMemoryDestination. agingRequiresTestedCycle
[SWS_DM_01577]	Behavior of not configured DiagnosticMemoryDestination. statusBitHandlingTestFailedSinceLastClear

▽

△

Number	Heading
[SWS_DM_01578]	Behavior of not configured DiagnosticMemoryDestination.clearDtcLimitation
[SWS_DM_01579]	Behavior of not configured DiagnosticMemoryDestination.memoryEntryStorageTrigger
[SWS_DM_01581]	Assigning a UDS request to a new Diagnostic Conversation in active default session
[SWS_DM_01582]	Healing counter increment
[SWS_DM_01583]	Resetting counter-based debouncing
[SWS_DM_01584]	Resetting time-based debouncing
[SWS_DM_01596]	Definition of API function ara::diag::OperationCycle::operator=
[SWS_DM_01597]	Definition of API function ara::diag::OperationCycle::operator=
[SWS_DM_01598]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_01599]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_01607]	Definition of API function ara::diag::Authentication::operator=
[SWS_DM_01608]	Definition of API function ara::diag::Authentication::operator=
[SWS_DM_01609]	Definition of API function ara::diag::Authentication::Authentication
[SWS_DM_01610]	Definition of API function ara::diag::Authentication::Authentication
[SWS_DM_01611]	Definition of API function ara::diag::DTCInformation::operator=
[SWS_DM_01612]	Definition of API function ara::diag::DTCInformation::operator=
[SWS_DM_01613]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_01614]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_01615]	Definition of API function ara::diag::SecurityAccess::operator=
[SWS_DM_01616]	Definition of API function ara::diag::SecurityAccess::operator=
[SWS_DM_01617]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_01618]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_01619]	Definition of API function ara::diag::EcuResetRequest::operator=
[SWS_DM_01620]	Definition of API function ara::diag::EcuResetRequest::operator=
[SWS_DM_01621]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01622]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01623]	Definition of API function ara::diag::Condition::operator=
[SWS_DM_01624]	Definition of API function ara::diag::Condition::operator=
[SWS_DM_01625]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_01626]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_01627]	Definition of API function ara::diag::Indicator::operator=
[SWS_DM_01628]	Definition of API function ara::diag::Indicator::operator=
[SWS_DM_01629]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_01630]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_01631]	Definition of API function ara::diag::DoIPActivationLine::operator=
[SWS_DM_01632]	Definition of API function ara::diag::DoIPActivationLine::operator=

▽

△

Number	Heading
[SWS_DM_01633]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_01634]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_01635]	Definition of API function ara::diag::GenericRoutine::operator=
[SWS_DM_01636]	Definition of API function ara::diag::GenericRoutine::operator=
[SWS_DM_01637]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_01638]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_01639]	Definition of API function ara::diag::DoIPGroupIdentification::operator=
[SWS_DM_01640]	Definition of API function ara::diag::DoIPGroupIdentification::operator=
[SWS_DM_01641]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroup Identification
[SWS_DM_01642]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroup Identification
[SWS_DM_01643]	Definition of API function ara::diag::DoIPPowerMode::operator=
[SWS_DM_01644]	Definition of API function ara::diag::DoIPPowerMode::operator=
[SWS_DM_01645]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_01646]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_01647]	Definition of API function ara::diag::DownloadService::operator=
[SWS_DM_01648]	Definition of API function ara::diag::DownloadService::operator=
[SWS_DM_01649]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01650]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01651]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01652]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01653]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01654]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01655]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01656]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01657]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01658]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01659]	Definition of API function {<namespace-list-data-element>}::{ <data-element-interface-name>}::operator=
[SWS_DM_01660]	Definition of API function {<namespace-list-data-element>}::{ <data-element-interface-name>}::operator=
[SWS_DM_01661]	Definition of API function {<namespace-list-data-element>}::{ <data-element-interface-name>}::{ <data-element-interface-name>}
[SWS_DM_01662]	Definition of API function {<namespace-list-data-element>}::{ <data-element-interface-name>}::{ <data-element-interface-name>}

▽

△

Number	Heading
[SWS_DM_01663]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::operator=
[SWS_DM_01664]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::operator=
[SWS_DM_01665]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}
[SWS_DM_01666]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name>}::{<data-identifier-interface-name>}
[SWS_DM_01667]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::operator=
[SWS_DM_01668]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::operator=
[SWS_DM_01669]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}
[SWS_DM_01670]	Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name>}::{<routine-interface-name>}
[SWS_DM_01671]	Definition of API function ara::diag::UploadService::operator=
[SWS_DM_01672]	Definition of API function ara::diag::UploadService::operator=
[SWS_DM_01673]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_01674]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_01675]	Definition of API function ara::diag::CommunicationControl::operator=
[SWS_DM_01676]	Definition of API function ara::diag::CommunicationControl::operator=
[SWS_DM_01677]	Definition of API function ara::diag::CommunicationControl::CommunicationControl
[SWS_DM_01678]	Definition of API function ara::diag::CommunicationControl::CommunicationControl
[SWS_DM_01679]	Definition of API function ara::diag::Event::operator=
[SWS_DM_01680]	Definition of API function ara::diag::Event::operator=
[SWS_DM_01681]	Definition of API function ara::diag::Event::Event
[SWS_DM_01682]	Definition of API function ara::diag::Event::Event
[SWS_DM_01683]	Definition of API function ara::diag::Monitor::operator=
[SWS_DM_01684]	Definition of API function ara::diag::Monitor::operator=
[SWS_DM_01685]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01686]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01687]	Definition of API function ara::diag::ServiceValidation::operator=
[SWS_DM_01688]	Definition of API function ara::diag::ServiceValidation::operator=
[SWS_DM_01689]	Definition of API function ara::diag::ServiceValidation::ServiceValidation

▽

△

Number	Heading
[SWS_DM_01690]	Definition of API function ara::diag::ServiceValidation::ServiceValidation
[SWS_DM_01694]	Definition of API class ara::diag::MultipleMonitor
[SWS_DM_01695]	Definition of API function ara::diag::MultipleMonitor::MultipleMonitor
[SWS_DM_01696]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01697]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01698]	Definition of API function ara::diag::MultipleMonitor::ReportMonitorAction
[SWS_DM_01699]	Definition of API function ara::diag::MultipleMonitor::Offer
[SWS_DM_01700]	Definition of API function ara::diag::MultipleMonitor::StopOffer
[SWS_DM_01701]	Definition of API type ara::diag::MonitorHandleType
[SWS_DM_01702]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01703]	Definition of API type ara::diag::EventHandleType
[SWS_DM_01704]	Definition of API class ara::diag::MultipleEvent
[SWS_DM_01705]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_01706]	Definition of API function ara::diag::MultipleEvent::~MultipleEvent
[SWS_DM_01707]	Definition of API function ara::diag::MultipleEvent::GetEventStatus
[SWS_DM_01708]	Definition of API function ara::diag::MultipleEvent::SetEventStatusChanged Notifier
[SWS_DM_01709]	Definition of API function ara::diag::MultipleEvent::GetDTCNumber
[SWS_DM_01710]	Definition of API function ara::diag::MultipleEvent::GetDebouncingStatus
[SWS_DM_01711]	Definition of API function ara::diag::MultipleEvent::GetFaultDetectionCounter
[SWS_DM_01726]	Definition of API type ara::diag::ConditionHandleType
[SWS_DM_01727]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_01728]	Definition of API function ara::diag::MultipleCondition::GetCondition
[SWS_DM_01729]	Definition of API function ara::diag::MultipleCondition::SetCondition
[SWS_DM_01730]	Definition of API class ara::diag::MultipleCondition
[SWS_DM_01731]	MultipleMonitor MonitorHandleType Generation
[SWS_DM_01732]	Invalid MonitorHandleType
[SWS_DM_01733]	MultipleEvent EventHandleType Generation
[SWS_DM_01734]	Invalid EventHandleType
[SWS_DM_01735]	MultipleCondition ConditionHandleType Generation
[SWS_DM_01736]	Invalid ConditionHandleType
[SWS_DM_01737]	Definition of API function ara::diag::MultipleCondition::~MultipleCondition
[SWS_DM_01739]	Authentication disabled
[SWS_DM_01740]	Initial enable condition state with notifier callback for disabled enable conditions
[SWS_DM_01741]	No more method calls after stop
[SWS_DM_01742]	Asynchronous UDS protocol start
[SWS_DM_01743]	Require a started UdsTransportProtocolHandler
[SWS_DM_01744]	Processing of ChannelReestablished

▽

△

Number	Heading
[SWS_DM_01745]	Behavior on failed transport protocol initialization
[SWS_DM_01746]	Required successful initialization
[SWS_DM_01747]	S3 timer value
[SWS_DM_01749]	Definition of API function ara::diag::EventStatusByte::operator=
[SWS_DM_01750]	Definition of API function ara::diag::EventStatusByte::operator=
[SWS_DM_01751]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01752]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01753]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01754]	Definition of API function ara::diag::EventStatusByte::IsNotSet
[SWS_DM_01755]	Definition of API function ara::diag::EventStatusByte::IsSet
[SWS_DM_01756]	Definition of API function ara::diag::EventStatusByte::IsPassedAndTested
[SWS_DM_01757]	Definition of API function ara::diag::EventStatusByte::IsFailedAndTested
[SWS_DM_01758]	Security Access Delay Timer on power up
[SWS_DM_01759]	DiagnosticDataElement serialization respecting the data element endianness for typed interfaces
[SWS_DM_01781]	(Re-)Identification of Clients by Token
[SWS_DM_01782]	(Re-)Identification of Clients by Identity
[SWS_DM_01783]	Locking only for authorized SOVD clients
[SWS_DM_01784]	Service Validation of SOVD Requests
[SWS_DM_01785]	Service Validation for SOVD Resource Collections
[SWS_DM_01786]	SOVD Online Capability Description role sensitivity
[SWS_DM_01787]	Realization of SOVD Read Configuration
[SWS_DM_01788]	Realization of SOVD Write Configuration
[SWS_DM_01789]	Realization of SOVD API Methods for Handling of Bulk Data
[SWS_DM_01790]	Realization of SOVD Bulk Data API Method for Retrieve List of all Bulk Data Categories
[SWS_DM_01791]	Realization of SOVD Read Bulk Data Meta Data
[SWS_DM_01792]	Realization of SOVD Download Bulk Data
[SWS_DM_01793]	Realization of SOVD Upload Bulk Data
[SWS_DM_01794]	Realization of SOVD Delete All Bulk Data Defined by Category
[SWS_DM_01795]	Realization of SOVD Delete Specific Bulk Data Resource
[SWS_DM_01796]	Realization of SOVD API Methods for Software Update
[SWS_DM_01797]	Realization of SOVD Retrieve List of All Updates
[SWS_DM_01798]	Realization of SOVD Get Details of Update
[SWS_DM_01799]	Realization of SOVD Automated Installation of an Update
[SWS_DM_01800]	Realization of SOVD Prepare Installation of an Update
[SWS_DM_01801]	Realization of SOVD Execute Installation of an Update
[SWS_DM_01802]	Realization of SOVD Get Status of an Update
[SWS_DM_01803]	Realization of SOVD Delete Update Package from an SOVD Server

▽

△

Number	Heading
[SWS_DM_01804]	Realization of SOVD Register an Update at the SOVD Server
[SWS_DM_01806]	Definition of API enum ara::diag::DiagSovdErrc
[SWS_DM_01807]	Definition of API class ara::diag::DiagSovdException
[SWS_DM_01808]	Definition of API function ara::diag::DiagSovdException::DiagSovdException
[SWS_DM_01809]	Definition of API class ara::diag::DiagSovdErrorDomain
[SWS_DM_01810]	Definition of API type ara::diag::DiagSovdErrorDomain::Errc
[SWS_DM_01811]	Definition of API type ara::diag::DiagSovdErrorDomain::Exception
[SWS_DM_01812]	Definition of API function ara::diag::DiagSovdErrorDomain::DiagSovdError Domain
[SWS_DM_01813]	Definition of API function ara::diag::DiagSovdErrorDomain::Name
[SWS_DM_01814]	Definition of API function ara::diag::DiagSovdErrorDomain::Message
[SWS_DM_01815]	Definition of API function ara::diag::DiagSovdErrorDomain::ThrowAs Exception
[SWS_DM_01816]	Definition of API function ara::diag::GetDiagSovdErrorDomain
[SWS_DM_01817]	Definition of API function ara::diag::MakeErrorCode
[SWS_DM_01818]	Definition of API variable ara::diag::MetaInfo::kSovd
[SWS_DM_01819]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorization Output::identity
[SWS_DM_01820]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorization Output::validUntil
[SWS_DM_01821]	Definition of API enum ara::diag::SovdRequestMethod
[SWS_DM_01822]	Definition of API class ara::diag::SovdServiceValidation
[SWS_DM_01823]	Definition of API function ara::diag::SovdServiceValidation::SovdService Validation
[SWS_DM_01824]	Definition of API function ara::diag::SovdServiceValidation::~SovdService Validation
[SWS_DM_01825]	Definition of API function ara::diag::SovdServiceValidation::Offer
[SWS_DM_01826]	Definition of API function ara::diag::SovdServiceValidation::StopOffer
[SWS_DM_01827]	Definition of API function ara::diag::SovdServiceValidation::Validate
[SWS_DM_01828]	Definition of API class ara::diag::HttpRedirect
[SWS_DM_01829]	Definition of API variable ara::diag::HttpRedirect::url
[SWS_DM_01830]	Definition of API class ara::diag::SovdBulkData::BulkDataMetaData Descriptor
[SWS_DM_01831]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::id
[SWS_DM_01832]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::mimetype
[SWS_DM_01833]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::name
[SWS_DM_01834]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::translation_id

▽

△

Number	Heading
[SWS_DM_01835]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::size
[SWS_DM_01836]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::file_name
[SWS_DM_01837]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::creation_date
[SWS_DM_01838]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::last_modified
[SWS_DM_01839]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash
[SWS_DM_01840]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash_algorithm
[SWS_DM_01841]	Definition of API class ara::diag::SovdBulkData
[SWS_DM_01842]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::deleted_items
[SWS_DM_01843]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::failed_items
[SWS_DM_01844]	Definition of API class ara::diag::SovdBulkData::DeletionError
[SWS_DM_01845]	Definition of API variable ara::diag::SovdBulkData::DeletionError::item
[SWS_DM_01846]	Definition of API variable ara::diag::SovdBulkData::DeletionError::error
[SWS_DM_01847]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01848]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01849]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01850]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01851]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01852]	Definition of API function ara::diag::SovdBulkData::~SovdBulkData
[SWS_DM_01853]	Definition of API function ara::diag::SovdBulkData::GetBulkDataMetaData
[SWS_DM_01854]	Definition of API function ara::diag::SovdBulkData::RequestBulkDataUpload
[SWS_DM_01855]	Definition of API function ara::diag::SovdBulkData::DeleteAllBulkData
[SWS_DM_01856]	Definition of API function ara::diag::SovdBulkData::DeleteSpecificBulkData
[SWS_DM_01857]	Definition of API function ara::diag::SovdBulkData::Offer
[SWS_DM_01858]	Definition of API function ara::diag::SovdBulkData::StopOffer
[SWS_DM_01860]	Definition of API class ara::diag::SovdConfiguration
[SWS_DM_01861]	Definition of API class ara::diag::SovdConfiguration::SovdConfigurationMeta Info
[SWS_DM_01862]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::size
[SWS_DM_01863]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::mimetype
[SWS_DM_01864]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01865]	Definition of API function ara::diag::SovdConfiguration::~SovdConfiguration

▽

△

Number	Heading
[SWS_DM_01866]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01867]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01868]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01869]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01870]	Definition of API function ara::diag::SovdConfiguration::RequestGet Configuration
[SWS_DM_01871]	Definition of API function ara::diag::SovdConfiguration::RequestPut Configuration
[SWS_DM_01872]	Definition of API function ara::diag::SovdConfiguration::Offer
[SWS_DM_01873]	Definition of API function ara::diag::SovdBulkData::RequestBulkData Download
[SWS_DM_01874]	Definition of API class ara::diag::SovdSwUpdate
[SWS_DM_01875]	Definition of API enum ara::diag::SovdUpdatePhase
[SWS_DM_01876]	Definition of API enum ara::diag::SovdUpdateStatus
[SWS_DM_01877]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageDetails
[SWS_DM_01878]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::id
[SWS_DM_01879]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::name
[SWS_DM_01881]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::translation_id
[SWS_DM_01882]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::automated
[SWS_DM_01883]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::origin
[SWS_DM_01884]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes
[SWS_DM_01885]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes_translation_id
[SWS_DM_01886]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity
[SWS_DM_01887]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity_translation_id
[SWS_DM_01888]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions
[SWS_DM_01889]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions_translation_id
[SWS_DM_01890]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::execution_conditions
[SWS_DM_01891]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::duration
[SWS_DM_01893]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::size

▽

△

Number	Heading
[SWS_DM_01894]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::updated_components
[SWS_DM_01895]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::affected_components
[SWS_DM_01896]	Definition of API class ara::diag::SovdSwUpdate::SubProgress
[SWS_DM_01898]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::entity
[SWS_DM_01899]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::status
[SWS_DM_01900]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::progress
[SWS_DM_01901]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::error
[SWS_DM_01902]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageStatus
[SWS_DM_01903]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::phase
[SWS_DM_01904]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::status
[SWS_DM_01905]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::progress
[SWS_DM_01906]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::subprogress
[SWS_DM_01907]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step
[SWS_DM_01908]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step_translation_id
[SWS_DM_01909]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::error
[SWS_DM_01910]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01911]	Definition of API function ara::diag::SovdSwUpdate::~SovdSwUpdate
[SWS_DM_01912]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01913]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01914]	Definition of API function ara::diag::SovdSwUpdate::operator=
[SWS_DM_01915]	Definition of API function ara::diag::SovdSwUpdate::operator=
[SWS_DM_01916]	Definition of API function ara::diag::SovdSwUpdate::GetAllUpdates
[SWS_DM_01917]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Details
[SWS_DM_01918]	Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackage Automated
[SWS_DM_01919]	Definition of API function ara::diag::SovdSwUpdate::PrepareUpdatePackage
[SWS_DM_01920]	Definition of API function ara::diag::SovdSwUpdate::ExecuteUpdatePackage
[SWS_DM_01921]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Status
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackage Registration

▽

△

Number	Heading
[SWS_DM_01924]	Definition of API function ara::diag::SovdSwUpdate::Offer
[SWS_DM_01925]	Definition of API function ara::diag::SovdSwUpdate::StopOffer
[SWS_DM_01926]	Definition of API class ara::diag::SovdBulkData::DeleteByCategoryResult
[SWS_DM_01927]	Definition of API class ara::diag::SovdAuthorization::ValidateAuthorization Output
[SWS_DM_01928]	SOVD method for logging
[SWS_DM_01929]	SOVD Error for non-reentrant software
[SWS_DM_01930]	Definition of API function ara::diag::SovdConfiguration::StopOffer
[SWS_DM_09010]	Definition of API function apext::diag::uds_transport::UdsMessage::~Uds Message
[SWS_DM_09012]	Definition of API function apext::diag::uds_transport::UdsMessage::Uds Message
[SWS_DM_09015]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::UdsTransportProtocolHandler
[SWS_DM_09016]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Handler::~UdsTransportProtocolHandler
[SWS_DM_09017]	Definition of API enum apext::diag::uds_transport::UdsTransportProtocol Handler::InitializationResult
[SWS_DM_09021]	Definition of API type apext::diag::uds_transport::UdsTransportProtocol Mgr::GlobalChannelIdentifier
[SWS_DM_09025]	Definition of API variable apext::diag::uds_transport::UdsTransportProtocol Handler::transportprotocolManager

Table E.31: Changed Specification Items in R24-11

E.10.3 Deleted Specification Items in R24-11

Number	Heading
[SWS_DM_00046]	Each Diagnostic Conversation has its own session resources
[SWS_DM_00047]	Each Diagnostic Conversation has its own security-level resources
[SWS_DM_00088]	ControlDTCSetting influence (freeze)
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00427]	Priority of a Diagnostic Conversation
[SWS_DM_00444]	Check Support of UDS service ControlDTCSetting (0x85) in active session
[SWS_DM_00445]	Check Support of UDS service ControlDTCSetting (0x85) on active security level
[SWS_DM_00451]	Definition of API type ara::diag::uds_transport::Priority
[SWS_DM_00452]	Definition of API type ara::diag::uds_transport::ProtocolKind
[SWS_DM_00510]	Namespace of Service header files

▽

△

Number	Heading
[SWS_DM_00511]	Implementation Types header files existence
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00513]	Implementation Types header file namespace
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00561]	Deployment of diagnostic PortInterfaces
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00850]	Default Service Interface for reading DiagnosticDataIdentifier
[SWS_DM_00881]	Enable condition influence on debouncing behavior (freeze)
[SWS_DM_00907]	Default Service Interface for writing DiagnosticDataIdentifier
[SWS_DM_01018]	ECUReset ara::diag::ResetRequestType check
[SWS_DM_01019]	Custom ara::diag::ResetRequestType processing
[SWS_DM_01100]	Debouncing Algorithm not fitting
[SWS_DM_01129]	Definition of API function ara::diag::Authentication::TransmitCertificate
[SWS_DM_01234]	Unexpected verifyCertificateUnidirectional from a different client
[SWS_DM_01239]	Unexpected verifyCertificateBidirectional from a different client
[SWS_DM_01273]	Namespace for typed UDS Service RoutineControl (0x31)
[SWS_DM_01274]	Namespace for typed DiagnosticDataIdentifier interface
[SWS_DM_01275]	Namespace for typed DiagnosticDataElements interface
[SWS_DM_01292]	Definition of API function ara::diag::DTCInformation::SetDtcSuppression
[SWS_DM_01293]	Definition of API function ara::diag::DTCInformation::GetDtcSuppression
[SWS_DM_01294]	Definition of API enum ara::diag::DTCInformation::DtcSuppressionType
[SWS_DM_01296]	Behavior of service ResponseOnEvent with subfunction onDTCStatusChange for suppressed DTCs
[SWS_DM_01297]	Behavior of SetNumberOfStoredEntriesNotifier() for suppressed DTCs
[SWS_DM_01298]	Behavior of SetDTCStatusChangedNotifier() for suppressed DTCs
[SWS_DM_01299]	Enabling of the notification SetDTCStatusChangedNotifier() for suppressed DTCs
[SWS_DM_01300]	Behavior of the ara::diag::DTCInformation class for suppressed DTCs
[SWS_DM_01301]	Behavior of ara::diag::Event class methods for suppressed DTCs
[SWS_DM_01302]	Behavior of Diagnostic Client services ClearDiagnosticInformation for suppressed DTCs inside a group
[SWS_DM_01303]	Behavior of Diagnostic Client services ClearDiagnosticInformation for suppressed DTCs
[SWS_DM_01304]	Behavior of services ReadDTCInformation for ExtendedData and SnapshotData for suppressed DTCs

▽

△

Number	Heading
[SWS_DM_01305]	Behavior of services ReadDTCInformation with DTC mask record for suppressed DTCs
[SWS_DM_01306]	Functionality of GetDtcSuppression()
[SWS_DM_01307]	Precondition for suppression
[SWS_DM_01308]	Functionality of SetDtcSuppression()
[SWS_DM_01407]	SOVD method Delete All Faults of an Entity without scope
[SWS_DM_01408]	SOVD method Delete All Faults of an Entity with scope
[SWS_DM_01575]	Configuration of permanent storage of "TestFailed" status bits
[SWS_DM_01580]	
[SWS_DM_01760]	Security events for Diagnostic Management
[SWS_DM_01761]	Reporting 'Request out of range' to IdsM
[SWS_DM_01762]	Reporting 'Sequence error' to IdsM
[SWS_DM_01763]	Reporting 'Data written using WriteDataByIdentifer' to IdsM
[SWS_DM_01764]	Reporting 'Writing invalid data is requested' to IdsM
[SWS_DM_01765]	Reporting 'Download sequence requested' to IdsM
[SWS_DM_01766]	Reporting 'ECU reset triggered ECUReset' to IdsM
[SWS_DM_01767]	Reporting 'Communication control switched off' to IdsM
[SWS_DM_01768]	Reporting 'DTC fault memory cleared' to IdsM
[SWS_DM_01769]	Reporting 'DTC setting switched off' to IdsM
[SWS_DM_01770]	Reporting 'successful authentication' to IdsM
[SWS_DM_01771]	Reporting 'certificate failed' to IdsM
[SWS_DM_01772]	Reporting 'authentication required' to IdsM
[SWS_DM_01773]	Reporting 'required time delay not expired for security access' to IdsM
[SWS_DM_01774]	Reporting 'number of attempts exceeded for security access' to IdsM
[SWS_DM_01775]	Reporting 'security access with invalid key' to IdsM
[SWS_DM_01776]	Reporting 'security unlocked successfully for security access' to IdsM
[SWS_DM_01777]	Reporting 'security access denied' to IdsM
[SWS_DM_01778]	Reporting 'incorrect message length or invalid format' to IdsM
[SWS_DM_01779]	Reporting 'service subfunction not supported' to IdsM
[SWS_DM_01780]	Reporting service not supported to IdsM
[SWS_DM_01805]	Definition SOVD ara::diag errors
[SWS_DM_01931]	
[SWS_DM_01932]	
[SWS_DM_01933]	
[SWS_DM_01934]	
[SWS_DM_01935]	
[SWS_DM_01936]	
[SWS_DM_01937]	
[SWS_DM_01938]	

▽

△

Number	Heading
[SWS_DM_01939]	
[SWS_DM_01940]	
[SWS_DM_01941]	
[SWS_DM_01942]	
[SWS_DM_01943]	
[SWS_DM_01944]	
[SWS_DM_01945]	
[SWS_DM_01946]	
[SWS_DM_01947]	
[SWS_DM_01948]	
[SWS_DM_01949]	

Table E.32: Deleted Specification Items in R24-11

E.10.4 Added Constraints in R24-11

Number	Heading
[SWS_DM_-CONSTR_-00397]	Configurable Namespace for Diagnostic Management

Table E.33: Added Constraints in R24-11

E.10.5 Changed Constraints in R24-11

Number	Heading
[SWS_DM_-CONSTR_-00059]	Restriction on supported DTC format
[SWS_DM_-CONSTR_-00082]	Restriction on the configuration of the DTC group GroupOfAllDTCs
[SWS_DM_-CONSTR_-00084]	Each DTC shall be assigned to an event memory destination
[SWS_DM_-CONSTR_-00168]	Required operation cycles for diagnostic events

▽

△

Number	Heading
[SWS_DM_-CONSTR_-00206]	Supported format for data identifier for VINDataIdentifier
[SWS_DM_-CONSTR_-00394]	`Internal DiagnosticDataElements` are read-only
[SWS_DM_-CONSTR_-00395]	Restriction on DEM-exclusive `DiagnosticDataElements`
[SWS_DM_-CONSTR_-00396]	Restriction on DCM-exclusive `DiagnosticDataElements`
[SWS_DM_-CONSTR_-00960]	No support for `DEM_AGINGCTR_UPCNT_FIRST_ACTIVE`
[SWS_DM_-CONSTR_-00961]	Limits of priority values

Table E.34: Changed Constraints in R24-11

E.10.6 Deleted Constraints in R24-11

none