

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	R25-11

Document Change History			
Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	• Native Handling of SOVD Data and Operations • Enhanced Snapshot Record Handling for SOVD Use Cases • Harmonized Use of SOVD and UDS Extended Data Records • Adapt DoIP Behavior Based on Configured DoIP Protocol Version • Add SecurityEvents for DoIP • Extend IAM with New SecurityEvents • Add Class Derivation Details and Semantic Constraints for C++ Data Type Mapping • Enhance Code Quality, Reliability, and Robustness





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 DoIPEExtension • 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

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	50
1.1	Diagnostic interface	50
1.2	AUTOSAR Diagnostic Extract Template (DEXT)	50
1.3	Software Cluster	50
1.3.1	Diagnostic Server	50
1.3.2	Diagnostic Managers external dependencies	53
2	Acronyms and Abbreviations	54
3	Related documentation	59
3.1	Input documents & related standards and norms	59
3.2	Further applicable specification	61
4	Constraints and assumptions	62
4.1	Known Limitations	62
4.1.1	UDS support	62
4.1.2	DoIP support	64
4.1.3	SOVD support	64
4.1.4	General	65
4.1.5	AP stabilization	65
5	Dependencies to other Functional Clusters	66
5.1	Provided Interfaces	66
5.2	Required Interfaces	66
6	Requirements Tracing	68
7	Functional specification	93
7.1	UDS Transport Layer	94
7.1.1	Support of UDS Transport Layer	95
7.1.1.1	Initialization, Starting and Stopping of a UDS TransportLayer	95
7.1.1.2	UDS message reception on a UDS TransportLayer	97
7.1.1.3	UDS message transmission on a UDS TransportLayer	101
7.1.1.4	Channel Notifications	102
7.1.2	DoIP	103
7.1.3	Dispatching of UDS Requests	110
7.2	SOVD Transport Layer	110
7.3	Diagnostic Server	111
7.3.1	Interaction between DM and applications	112
7.3.1.1	MetaInfo class	113
7.3.1.1.1	Provision of MetaInfo	114
7.3.1.1.2	Lifetime of MetaInfo	116
7.3.1.2	Concurrency of ara::diag interfaces	116
7.3.1.3	Caching of application calls	116

7.3.2 Diagnostic Communication Management	117
7.3.2.1 Diagnostic Conversations	117
7.3.2.1.1 Multiple Client Handling	118
7.3.2.1.2 Life-cycle of a Diagnostic Conversation	121
7.3.2.1.3 Diagnostic Conversation Service Interface	121
7.3.2.2 Assignment of UDS requests to Diagnostic Conversations	123
7.3.2.2.1 Prioritization	125
7.3.2.2.2 Replacement of Diagnostic Conversations and initial values	127
7.3.2.2.3 Refusal of incoming diagnostic request	128
7.3.2.3 Handling Authentication State and DynamicAccessLists	128
7.3.2.3.1 ExternalAuthentication	128
7.3.2.3.2 ClientAuthentication	129
7.3.2.3.3 ClientAuthenticationHandle	132
7.3.2.3.4 DynamicAccessList Creation and Update	133
7.3.2.4 Diagnostic Service Authentication checks	134
7.3.2.5 UDS request Validation/Verification	136
7.3.2.5.1 UDS request format checks	137
7.3.2.5.2 Supported service checks	137
7.3.2.5.3 Session and Security Checks	138
7.3.2.5.4 Manufacturer and Supplier Permission Checks and Confirmation	139
7.3.2.5.5 Condition checks	141
7.3.2.6 UDS response handling	142
7.3.2.6.1 NRCs created by application	143
7.3.2.7 Keep track of active non-default sessions	144
7.3.2.8 UDS service processing	145
7.3.2.8.1 Supported UDS Services	145
7.3.2.8.2 Common service processing items	146
7.3.2.8.3 UDS service serialization	147
7.3.2.8.4 Service 0x10 – DiagnosticSessionControl	147
7.3.2.8.5 Service 0x11 – ECUReset	148
7.3.2.8.5.1 Service 0x11 – ECUReset – RapidPowerShutdown sub function	149
7.3.2.8.5.2 Service 0x11 – ECUReset – all other sub functions	149
7.3.2.8.6 Service 0x14 – ClearDiagnosticInformation	151
7.3.2.8.6.1 Clearing user-defined fault memory	153
7.3.2.8.7 Service 0x19 – ReadDTCInformation	154
7.3.2.8.7.1 SF 0x01 – reportNumberOfDTCByStatusMask	155
7.3.2.8.7.2 SF 0x02 – reportDTCByStatusMask	156
7.3.2.8.7.3 SF 0x03 – reportDTCsSnapshotIdentification	156
7.3.2.8.7.4 SF 0x04 – reportDTCsSnapshotRecordByDTCNumber	156
7.3.2.8.7.5 SF 0x06 – reportDTCExtDataRecordByDTCNumber	157

7.3.2.8.7.6	SF 0x07 – reportNumberOfDTCBySeverity-MaskRecord	157
7.3.2.8.7.7	SF 0x0A – reportSupportedDTC	157
7.3.2.8.7.8	SF 0x14 – reportDTCFaultDetectionCounter	158
7.3.2.8.7.9	SF 0x17 – reportUserDefMemoryDTCByStatus-Mask	158
7.3.2.8.7.10	SF 0x18 – reportUserDefMemoryDTCSnapshotRecordByDTCNumber	158
7.3.2.8.7.11	SF 0x19 – reportUserDefMemoryDTCExt-DataRecordByDTCNumber	159
7.3.2.8.8	Service 0x22 – ReadDataByIdentifier	159
7.3.2.8.9	Service 0x27 – SecurityAccess	161
7.3.2.8.10	Service 0x28 – CommunicationControl	164
7.3.2.8.11	Service 0x29 – Authentication	164
7.3.2.8.11.1	Service 0x29 – deAuthenticate (0x00)	166
7.3.2.8.11.2	Service 0x29 – verifyCertificateUnidirectional (0x01)	166
7.3.2.8.11.3	Service 0x29 – verifyCertificateBidirectional (0x02)	167
7.3.2.8.11.4	Service 0x29 – proofOfOwnership (0x03)	168
7.3.2.8.11.5	Service 0x29 – transmitCertificate (0x04)	169
7.3.2.8.11.6	Service 0x29 – authenticationConfiguration (0x08)	170
7.3.2.8.12	Service 0x2A – ReadDataByPeriodicIdentifier	170
7.3.2.8.13	Service 0x2C – DynamicallyDefineDataIdentifier	175
7.3.2.8.14	Service 0x2E – WriteDataByIdentifier	177
7.3.2.8.15	Service 0x31 – RoutineControl	178
7.3.2.8.16	Service 0x34 – RequestDownload	181
7.3.2.8.17	Service 0x35 – RequestUpload	181
7.3.2.8.18	Service 0x36 – TransferData	181
7.3.2.8.19	Service 0x37 – RequestTransferExit	182
7.3.2.8.20	Service 0x38 – RequestFileTransfer	183
7.3.2.8.21	Service 0x3E – TesterPresent	186
7.3.2.8.22	Service 0x85 – ControlDTCSetting	186
7.3.2.8.23	Service 0x86 – ResponseOnEvent	188
7.3.2.8.24	Custom Diagnostic Services	191
7.3.2.9	Cancellation of a Diagnostic Conversation	191
7.3.3	Diagnostic SOVD Management	193
7.3.3.1	SOVD Conversations and UDS Interplay	193
7.3.3.2	SOVD Request Validation and Verification	194
7.3.3.2.1	SOVD – Authorization	194
7.3.3.2.2	SOVD – Locks	195
7.3.3.2.3	SOVD – Validation of Environmental Conditions	196
7.3.3.2.4	SOVD – Service Validation of Requests	196
7.3.3.3	SOVD Data Serialization & Data Conversion	197
7.3.3.3.1	SOVD Data – Serialization	197
7.3.3.3.2	SOVD Data – Conversion	197

7.3.3.4	Standardized SOVD APIs	199
7.3.3.4.1	SOVD resource docs	199
7.3.3.4.2	SOVD resource version-info	200
7.3.3.4.3	SOVD data-categories	201
7.3.3.4.4	SOVD data-groups	202
7.3.3.4.5	SOVD method Query entity data	202
7.3.3.4.6	SOVD method Locks	203
7.3.3.4.7	SOVD method Logging	204
7.3.3.5	Configurable SOVD APIs	204
7.3.3.5.1	SOVD API – Data Access	204
7.3.3.5.2	SOVD API – Configuration	206
7.3.3.5.2.1	Configuration – Explicit Configurations	207
7.3.3.5.3	SOVD API – Data Lists	207
7.3.3.5.4	SOVD API – Faults	208
7.3.3.5.5	SOVD API – Operations	212
7.3.3.5.5.1	Operations – Query for operations from an Entity	213
7.3.3.5.5.2	Operations – Query for executions of an operation	214
7.3.3.5.5.3	Operations – Proximity Management	214
7.3.3.5.5.4	Operations – Retrieve details of an operation	216
7.3.3.5.5.5	Operations – Start execution of an operation	217
7.3.3.5.5.6	Operations – Terminate execution of an operation	218
7.3.3.5.5.7	Operations – Read the status of an operation execution	218
7.3.3.5.6	SOVD API – Modes	219
7.3.3.5.6.1	Modes – CommunicationControl	220
7.3.3.5.6.2	Modes – ControlDTCSetting	220
7.3.3.5.7	SOVD API – Bulk Data	221
7.3.3.5.8	SOVD API – Software Update	222
7.3.4	Diagnostic Event Management	225
7.3.4.1	Diagnostic Events	225
7.3.4.1.1	Event Definition	225
7.3.4.1.2	Monitors	227
7.3.4.1.3	Event Combination	232
7.3.4.1.3.1	Combination On Storage	234
7.3.4.1.3.2	Combination On Retrieval	235
7.3.4.1.4	EnableConditions	236
7.3.4.1.5	Debouncing	237
7.3.4.1.5.1	Counter-based debouncing	239
7.3.4.1.5.2	Time-based debouncing	243
7.3.4.1.5.3	Monitor-internal debouncing	246
7.3.4.1.5.4	Debounce algorithm reset	247
7.3.4.1.5.5	Dependencies to enable conditions	248
7.3.4.1.5.6	Dependencies to UDS service 0x85 ControlDTC-Settings	248

7.3.4.1.6	Event Status processing	248
7.3.4.1.7	Event status change notifications	250
7.3.4.1.8	Event occurrence	250
7.3.4.2	Condition Mangement	251
7.3.4.3	Operation Cycles Management	252
7.3.4.4	Event memory	253
7.3.4.4.1	DTC Introduction	254
7.3.4.4.1.1	Format	255
7.3.4.4.1.2	Groups	255
7.3.4.4.1.3	Priority	256
7.3.4.4.2	UDS DTC Status	256
7.3.4.4.2.1	Status processing	256
7.3.4.4.2.2	UDS DTC Status change notifications	258
7.3.4.4.2.3	Indicators	258
7.3.4.4.2.4	User controlled WarningIndicatorRequest-bit	260
7.3.4.4.3	Destination	261
7.3.4.4.4	DTC related data	261
7.3.4.4.4.1	Triggering for data storage	262
7.3.4.4.4.2	Storage of snapshot record data	262
7.3.4.4.4.3	Storage of extended data	265
7.3.4.4.4.4	Internal statistical data elements in EDRs	266
7.3.4.4.5	Clearing DTCs	272
7.3.4.4.5.1	Locking of the DTC clearing process by a client	273
7.3.4.4.5.2	ClearConditions	274
7.3.4.4.5.3	DTC clearing triggered by application	275
7.3.4.4.6	Aging	276
7.3.4.4.7	NumberOfStoredEntries	278
7.3.4.4.8	Active / Passive Status of Events	278
7.3.4.4.9	Event memory overflow	279
7.3.4.4.10	Event memory entry displacement	280
7.3.4.4.11	Reporting order of event memory entries	285
7.3.5	Required Configuration	285
7.3.6	Diagnostic Data Management	286
7.3.6.1	Internal and External Diagnostic Data Elements	286
7.3.6.2	Reading and Writing Diagnostic Data Identifier	290
7.3.6.2.1	Supported Diagnostic Mappings	290
7.3.6.2.2	Reading Diagnostic Data Identifier	290
7.3.6.2.3	Writing Diagnostic Data Identifier	291
7.4	Functional cluster life-cycle	292
7.4.1	Startup	292
7.4.2	Shutdown	293
7.4.3	Daemon crash	293
7.5	Reporting	293

7.5.1	Security Events	293
7.5.2	Log Messages	309
7.5.3	Violation Messages	310
7.5.4	Production Errors	312
7.6	Access Control	312
8	API specification	314
8.1	PortInterface to API class binding	314
8.2	Diagnostic Communication-related APIs	317
8.3	Header: ara/diag/authentication.h	317
8.3.1	Class: Authentication	317
8.3.1.1	Public Member Functions	318
8.3.1.1.1	Special Member Functions	318
8.3.1.1.1.1	Move Constructor	318
8.3.1.1.1.2	Move Assignment Operator	318
8.3.1.1.1.3	Destructor	318
8.3.1.1.2	Constructors	319
8.3.1.1.2.1	Authentication	319
8.3.1.1.2.2	Authentication	320
8.3.1.1.3	Member Functions	320
8.3.1.1.3.1	Offer	320
8.3.1.1.3.2	StopOffer	321
8.3.1.1.3.3	VerifyCertificateBidirectional	321
8.3.1.1.3.4	VerifyCertificateUnidirectional	322
8.3.1.1.3.5	VerifyOwnership	323
8.3.1.1.3.6	operator=	324
8.4	Header: ara/diag/cancellation_handler.h	325
8.4.1	Class: CancellationHandler	325
8.4.1.1	Public Member Types	325
8.4.1.1.1	Type Alias: CancellationHandlerSetNotifier	325
8.4.1.2	Public Member Functions	326
8.4.1.2.1	Special Member Functions	326
8.4.1.2.1.1	Copy Constructor	326
8.4.1.2.1.2	Default Constructor	326
8.4.1.2.1.3	Move Constructor	327
8.4.1.2.1.4	Move Assignment Operator	327
8.4.1.2.2	Member Functions	328
8.4.1.2.2.1	IsCanceled	328
8.4.1.2.2.2	SetNotifier	328
8.4.1.2.2.3	operator=	329
8.5	Header: ara/diag/client_authentication.h	329
8.5.1	Class: ClientAuthentication	329
8.5.1.1	Public Member Types	330
8.5.1.1.1	Type Alias: ClientAuthenticationSetNotifier	330

8.5.1.1.2	Type Alias: DiagnosticAuthRole	330
8.5.1.1.3	Enumeration: DiagnosticAuthState	331
8.5.1.2	Public Member Functions	331
8.5.1.2.1	Special Member Functions	331
8.5.1.2.1.1	Move Constructor	331
8.5.1.2.1.2	Destructor	332
8.5.1.2.2	Constructors	332
8.5.1.2.2.1	ClientAuthentication	332
8.5.1.2.3	Member Functions	333
8.5.1.2.3.1	Authenticate	333
8.5.1.2.3.2	GetState	333
8.5.1.2.3.3	OverrideDefaultRoles	334
8.5.1.2.3.4	SetNotifier	334
8.5.1.2.3.5	operator=(ClientAuthentication&&) &	335
8.5.1.2.3.6	operator=(ClientAuthentication const&)	335
8.6	Header: ara/diag/client_authentication_handle.h	336
8.6.1	Class: ClientAuthenticationHandle	336
8.6.1.1	Public Member Functions	336
8.6.1.1.1	Special Member Functions	336
8.6.1.1.1.1	Default Constructor	336
8.6.1.1.1.2	Move Constructor	337
8.6.1.1.1.3	Destructor	337
8.6.1.1.2	Constructors	338
8.6.1.1.2.1	ClientAuthenticationHandle	338
8.6.1.1.3	Member Functions	338
8.6.1.1.3.1	Append	338
8.6.1.1.3.2	Refresh	339
8.6.1.1.3.3	Revoke	339
8.6.1.1.3.4	Set	340
8.6.1.1.3.5	operator=(ClientAuthenticationHandle&&) &	340
8.6.1.1.3.6	operator=(ClientAuthenticationHandle const&)	341
8.7	Header: ara/diag/communication_control.h	341
8.7.1	Class: CommunicationControl	341
8.7.1.1	Public Member Functions	342
8.7.1.1.1	Special Member Functions	342
8.7.1.1.1.1	Move Constructor	342
8.7.1.1.1.2	Move Assignment Operator	342
8.7.1.1.1.3	Destructor	343
8.7.1.1.2	Constructors	343
8.7.1.1.2.1	CommunicationControl	343
8.7.1.1.2.2	CommunicationControl	344
8.7.1.1.3	Member Functions	344
8.7.1.1.3.1	CommCtrlRequest	344

8.7.1.1.3.2	Offer	345
8.7.1.1.3.3	StopOffer	346
8.7.1.1.3.4	operator=	346
8.7.2	Struct: ComCtrlRequestParamsType	346
8.7.2.1	Public Member Variables	347
8.7.2.1.1	communicationType	347
8.7.2.1.2	controlType	347
8.7.2.1.3	nodeIdentificationNumber	348
8.8	Header: ara/diag/concurrency.h	348
8.8.1	Non-Member Types	348
8.8.1.1	Enumeration: ConcurrencyType	348
8.8.2	Struct: DataIdentifierConcurrencyType	349
8.8.2.1	Public Member Variables	349
8.8.2.1.1	read	349
8.8.2.1.2	readWrite	350
8.8.2.1.3	write	350
8.8.3	Struct: SovdContentConcurrencyType	351
8.8.3.1	Public Member Variables	351
8.8.3.1.1	read	351
8.8.3.1.2	readWrite	352
8.8.3.1.3	write	352
8.9	Header: ara/diag/conversation.h	353
8.9.1	Non-Member Types	353
8.9.1.1	Enumeration: ActivityStatusType	353
8.9.1.2	Type Alias: SecurityLevelType	353
8.9.1.3	Type Alias: SessionControlType	354
8.9.2	Global Variables	354
8.9.2.1	kDefaultSession	354
8.9.2.2	kExtendedDiagnosticSession	354
8.9.2.3	kLocked	355
8.9.2.4	kProgrammingSession	355
8.9.2.5	kSafetySystemDiagnosticSession	356
8.9.3	Class: Conversation	356
8.9.3.1	Public Member Types	357
8.9.3.1.1	Type Alias: ActivityStatusTypeNotifier	357
8.9.3.1.2	Type Alias: SecurityLevelTypeNotifier	357
8.9.3.1.3	Type Alias: SessionControlTypeNotifier	358
8.9.3.2	Public Member Functions	358
8.9.3.2.1	Special Member Functions	358
8.9.3.2.1.1	Move Constructor	358
8.9.3.2.1.2	Copy Constructor	359
8.9.3.2.1.3	Copy Assignment Operator	359
8.9.3.2.1.4	Move Assignment Operator	359

8.9.3.2.2	Member Functions	360
8.9.3.2.2.1	GetActivityStatus	360
8.9.3.2.2.2	GetAllConversations	360
8.9.3.2.2.3	GetConversation	361
8.9.3.2.2.4	GetConversationIdentifier	361
8.9.3.2.2.5	GetCurrentActiveConversations	362
8.9.3.2.2.6	GetDiagnosticSecurityLevel	362
8.9.3.2.2.7	GetDiagnosticSecurityLevelShortName	363
8.9.3.2.2.8	GetDiagnosticSession	363
8.9.3.2.2.9	GetDiagnosticSessionShortName	364
8.9.3.2.2.10	ResetToDefaultSession	364
8.9.3.2.2.11	SetActivityNotifier	365
8.9.3.2.2.12	SetDiagnosticSessionNotifier	365
8.9.3.2.2.13	SetSecurityLevelNotifier	366
8.9.4	Struct: ConversationIdentifierType	366
8.10	Header: ara/diag/data_transfer.h	367
8.10.1	Non-Member Types	367
8.10.1.1	Enumeration: DataTransferExitType	367
8.10.2	Class: DataTransferReadByPullHandler	367
8.10.2.1	Public Member Functions	368
8.10.2.1.1	Special Member Functions	368
8.10.2.1.1.1	Move Constructor	368
8.10.2.1.1.2	Destructor	368
8.10.2.1.2	Constructors	369
8.10.2.1.2.1	DataTransferReadByPullHandler	369
8.10.2.1.3	Member Functions	369
8.10.2.1.3.1	ExitRead	369
8.10.2.1.3.2	Read	370
8.10.2.1.3.3	operator=(DataTransferReadByPullHandler&&) &	371
8.10.2.1.3.4	operator=(DataTransferReadByPullHandler const&)	372
8.10.2.2	Protected Member Functions	372
8.10.2.2.1	Special Member Functions	372
8.10.2.2.1.1	Default Constructor	372
8.10.3	Class: DataTransferReadByPushHandler	373
8.10.3.1	Public Member Functions	373
8.10.3.1.1	Special Member Functions	373
8.10.3.1.1.1	Move Constructor	373
8.10.3.1.1.2	Destructor	374
8.10.3.1.2	Constructors	374
8.10.3.1.2.1	DataTransferReadByPushHandler	374
8.10.3.1.3	Member Functions	375
8.10.3.1.3.1	ExitRead	375
8.10.3.1.3.2	Read	376

8.10.3.1.3.3	operator=(DataTransferReadByPushHandler&&) &	377
8.10.3.1.3.4	operator=(DataTransferReadByPushHandler const&)	377
8.10.3.2	Protected Member Functions	378
8.10.3.2.1	Special Member Functions	378
8.10.3.2.1.1	Default Constructor	378
8.10.4	Class: DataTransferReadSession	378
8.10.4.1	Public Member Functions	379
8.10.4.1.1	Special Member Functions	379
8.10.4.1.1.1	Default Constructor	379
8.10.4.1.1.2	Move Constructor	379
8.10.4.1.1.3	Destructor	380
8.10.4.1.2	Constructors	380
8.10.4.1.2.1	DataTransferReadSession	380
8.10.4.1.3	Member Functions	381
8.10.4.1.3.1	operator=(DataTransferReadSession const&)	381
8.10.4.1.3.2	operator=(DataTransferReadSession&&) &	381
8.10.5	Class: DataTransferReadSharedDataHandler	382
8.10.5.1	Public Member Functions	382
8.10.5.1.1	Special Member Functions	382
8.10.5.1.1.1	Default Constructor	382
8.10.5.1.1.2	Move Constructor	383
8.10.5.1.1.3	Destructor	383
8.10.5.1.2	Constructors	384
8.10.5.1.2.1	DataTransferReadSharedDataHandler	384
8.10.5.1.3	Member Functions	384
8.10.5.1.3.1	ExitRead	384
8.10.5.1.3.2	Read	385
8.10.5.1.3.3	operator=(DataTransferReadSharedDataHandler&&) &	386
8.10.5.1.3.4	operator=(DataTransferReadSharedDataHandler const&)	387
8.10.6	Class: DataTransferWriteHandler	387
8.10.6.1	Public Member Functions	388
8.10.6.1.1	Special Member Functions	388
8.10.6.1.1.1	Move Constructor	388
8.10.6.1.1.2	Destructor	388
8.10.6.1.2	Constructors	389
8.10.6.1.2.1	DataTransferWriteHandler	389
8.10.6.1.3	Member Functions	389
8.10.6.1.3.1	ExitWrite	389
8.10.6.1.3.2	Write	390
8.10.6.1.3.3	operator=(DataTransferWriteHandler&&) &	391
8.10.6.1.3.4	operator=(DataTransferWriteHandler const&)	391
8.10.6.2	Protected Member Functions	392

8.10.6.2.1 Special Member Functions	392
8.10.6.2.1.1 Default Constructor	392
8.10.7 Class: DataTransferWriteSession	392
8.10.7.1 Public Member Functions	393
8.10.7.1.1 Special Member Functions	393
8.10.7.1.1.1 Default Constructor	393
8.10.7.1.1.2 Move Constructor	393
8.10.7.1.1.3 Destructor	394
8.10.7.1.2 Constructors	394
8.10.7.1.2.1 DataTransferWriteSession	394
8.10.7.1.3 Member Functions	395
8.10.7.1.3.1 operator=(DataTransferWriteSession const&)	395
8.10.7.1.3.2 operator=(DataTransferWriteSession&&) &	395
8.11 Header: ara/diag/diag_error_domain.h	396
8.11.1 Non-Member Types	396
8.11.1.1 Enumeration: DiagErrc	396
8.11.1.2 Enumeration: DiagOfferErrc	397
8.11.1.3 Enumeration: DiagReportingErrc	398
8.11.2 Non-Member Functions	398
8.11.2.1 Other	398
8.11.2.1.1 GetDiagDomain	398
8.11.2.1.2 GetDiagOfferDomain	399
8.11.2.1.3 GetDiagReportingDomain	399
8.11.2.1.4 MakeErrorCode	400
8.11.2.1.5 MakeErrorCode	400
8.11.2.1.6 MakeErrorCode	401
8.11.3 Class: DiagErrorDomain	401
8.11.3.1 Public Member Types	402
8.11.3.1.1 Type Alias: Errc	402
8.11.3.1.2 Type Alias: Exception	402
8.11.3.2 Public Member Functions	403
8.11.3.2.1 Special Member Functions	403
8.11.3.2.1.1 Default Constructor	403
8.11.3.2.2 Member Functions	403
8.11.3.2.2.1 Message	403
8.11.3.2.2.2 Name	404
8.11.3.2.2.3 ThrowAsException	404
8.11.4 Class: DiagException	405
8.11.4.1 Public Member Functions	405
8.11.4.1.1 Constructors	405
8.11.4.1.1.1 DiagException	405
8.11.5 Class: DiagOfferErrorDomain	406
8.11.5.1 Public Member Types	406

8.11.5.1.1 Type Alias: Errc	406
8.11.5.1.2 Type Alias: Exception	407
8.11.5.2 Public Member Functions	407
8.11.5.2.1 Special Member Functions	407
8.11.5.2.1.1 Default Constructor	407
8.11.5.2.2 Member Functions	408
8.11.5.2.2.1 Message	408
8.11.5.2.2.2 Name	408
8.11.5.2.2.3 ThrowAsException	409
8.11.6 Class: DiagOfferException	409
8.11.6.1 Public Member Functions	410
8.11.6.1.1 Constructors	410
8.11.6.1.1.1 DiagOfferException	410
8.11.7 Class: DiagReportingErrorDomain	410
8.11.7.1 Public Member Types	411
8.11.7.1.1 Type Alias: Errc	411
8.11.7.1.2 Type Alias: Exception	411
8.11.7.2 Public Member Functions	412
8.11.7.2.1 Special Member Functions	412
8.11.7.2.1.1 Default Constructor	412
8.11.7.2.2 Member Functions	412
8.11.7.2.2.1 Message	412
8.11.7.2.2.2 Name	413
8.11.7.2.2.3 ThrowAsException	413
8.11.8 Class: DiagReportingException	414
8.11.8.1 Public Member Functions	414
8.11.8.1.1 Constructors	414
8.11.8.1.1.1 DiagReportingException	414
8.12 Header: ara/diag/diag_uds_nrc_error_domain.h	415
8.12.1 Non-Member Types	415
8.12.1.1 Enumeration: DiagUdsNrcErrc	415
8.12.2 Non-Member Functions	422
8.12.2.1 Other	422
8.12.2.1.1 GetDiagUdsNrcDomain	422
8.12.2.1.2 MakeErrorCode	423
8.12.3 Class: DiagUdsNrcErrorDomain	423
8.12.3.1 Public Member Types	424
8.12.3.1.1 Type Alias: Errc	424
8.12.3.1.2 Type Alias: Exception	424
8.12.3.2 Public Member Functions	425
8.12.3.2.1 Special Member Functions	425
8.12.3.2.1.1 Default Constructor	425
8.12.3.2.2 Member Functions	425

8.12.3.2.2.1	Message	425
8.12.3.2.2.2	Name	426
8.12.3.2.2.3	ThrowAsException	426
8.12.4	Class: DiagUdsNrcException	427
8.12.4.1	Public Member Functions	427
8.12.4.1.1	Constructors	427
8.12.4.1.1.1	DiagUdsNrcException	427
8.13	Header: ara/diag/diagnostic_service_dynamic_access_list.h	428
8.13.1	Class: DiagnosticServiceDynamicAccessList	428
8.13.1.1	Public Member Functions	428
8.13.1.1.1	Special Member Functions	428
8.13.1.1.1.1	Default Constructor	428
8.13.1.1.1.2	Move Constructor	429
8.13.1.1.1.3	Destructor	429
8.13.1.1.2	Constructors	430
8.13.1.1.2.1	DiagnosticServiceDynamicAccessList	430
8.13.1.1.3	Member Functions	430
8.13.1.1.3.1	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::ByteString)	430
8.13.1.1.3.2	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::Byte)	431
8.13.1.1.3.3	Reserve	431
8.13.1.1.3.4	operator=(DiagnosticServiceDynamicAccessList&&) &	432
8.13.1.1.3.5	operator=(DiagnosticServiceDynamicAccessList const&) &	432
8.14	Header: ara/diag/download.h	433
8.14.1	Class: DownloadService	433
8.14.1.1	Public Member Functions	433
8.14.1.1.1	Special Member Functions	433
8.14.1.1.1.1	Move Constructor	433
8.14.1.1.1.2	Move Assignment Operator	434
8.14.1.1.1.3	Destructor	434
8.14.1.1.2	Constructors	435
8.14.1.1.2.1	DownloadService(const ara::core::InstanceSpecifier&, ConcurrencyType)	435
8.14.1.1.2.2	DownloadService(DownloadService&)	435
8.14.1.1.3	Member Functions	436
8.14.1.1.3.1	DownloadData	436
8.14.1.1.3.2	Offer	437
8.14.1.1.3.3	RequestDownload	437
8.14.1.1.3.4	RequestDownloadExit	438
8.14.1.1.3.5	StopOffer	439
8.14.1.1.3.6	operator=	439

8.14.2 Struct: OperationOutput	440
8.14.2.1 Public Member Variables	440
8.14.2.1.1 responseData	440
8.15 Header: ara/diag/dynamic_access_list_diag_service_builder.h	441
8.15.1 Class: DynamicAccessListDiagServiceBuilder	441
8.15.1.1 Public Member Types	441
8.15.1.1.1 Type Alias: Byte	441
8.15.1.1.2 Type Alias: ByteString	442
8.15.1.2 Public Member Functions	442
8.15.1.2.1 Special Member Functions	442
8.15.1.2.1.1 Move Constructor	442
8.15.1.2.1.2 Destructor	443
8.15.1.2.2 Constructors	443
8.15.1.2.2.1 DynamicAccessListDiagServiceBuilder(V, ara::core::Vector<std::uint8_t>&)	443
8.15.1.2.2.2 DynamicAccessListDiagServiceBuilder(Dynamic AccessListDiagServiceBuilder const&)	444
8.15.1.2.3 Member Functions	444
8.15.1.2.3.1 Add(ByteRange)	444
8.15.1.2.3.2 Add(Byte)	445
8.15.1.2.3.3 Add(ByteString)	445
8.15.1.2.3.4 Any	446
8.15.1.2.3.5 Build	446
8.15.1.2.3.6 EndsWith(ByteRange)	447
8.15.1.2.3.7 EndsWith(Byte)	447
8.15.1.2.3.8 operator=(DynamicAccessListDiagServiceBuilder&&)	448
8.15.1.2.3.9 operator=(DynamicAccessListDiagServiceBuilder const&)	448
8.15.2 Class: ByteRange	448
8.15.2.1 Public Member Functions	449
8.15.2.1.1 Special Member Functions	449
8.15.2.1.1.1 Move Constructor	449
8.15.2.1.1.2 Destructor	449
8.15.2.1.2 Constructors	450
8.15.2.1.2.1 ByteRange	450
8.15.2.1.2.2 ByteRange	450
8.15.2.1.3 Member Functions	451
8.15.2.1.3.1 GetMax	451
8.15.2.1.3.2 GetMin	451
8.15.2.1.3.3 operator=	452
8.15.2.1.3.4 operator=	452
8.16 Header: ara/diag/ecu_reset_request.h	453
8.16.1 Non-Member Types	453
8.16.1.1 Type Alias: ResetRequestType	453

8.16.2 Global Variables	453
8.16.2.1 kCustomReset	453
8.16.2.2 kHardReset	454
8.16.2.3 kKeyOffOnReset	454
8.16.2.4 kSoftReset	454
8.16.3 Class: EcuResetRequest	455
8.16.3.1 Public Member Functions	455
8.16.3.1.1 Special Member Functions	455
8.16.3.1.1.1 Move Constructor	455
8.16.3.1.1.2 Move Assignment Operator	456
8.16.3.1.1.3 Destructor	456
8.16.3.1.2 Constructors	457
8.16.3.1.2.1 EcuResetRequest(const ara::core::InstanceSpeci- fier&)	457
8.16.3.1.2.2 EcuResetRequest(EcuResetRequest&)	457
8.16.3.1.3 Member Functions	458
8.16.3.1.3.1 EnableRapidShutdown	458
8.16.3.1.3.2 ExecuteReset	458
8.16.3.1.3.3 Offer	459
8.16.3.1.3.4 RequestReset	459
8.16.3.1.3.5 StopOffer	460
8.16.3.1.3.6 operator=	461
8.17 Header: ara/diag/external_authentication.h	461
8.17.1 Class: ExternalAuthentication	461
8.17.1.1 Public Member Types	462
8.17.1.1.1 Type Alias: Address	462
8.17.1.2 Public Member Functions	462
8.17.1.2.1 Special Member Functions	462
8.17.1.2.1.1 Move Constructor	462
8.17.1.2.1.2 Destructor	463
8.17.1.2.2 Constructors	463
8.17.1.2.2.1 ExternalAuthentication(ara::core::InstanceSpecifier)	463
8.17.1.2.2.2 ExternalAuthentication(ExternalAuthentication const&)	464
8.17.1.2.3 Member Functions	464
8.17.1.2.3.1 Get(Address)	464
8.17.1.2.3.2 Get(const MetaInfo&)	465
8.17.1.2.3.3 GetAll	465
8.17.1.2.3.4 operator=(ExternalAuthentication const&)	466
8.17.1.2.3.5 operator=(ExternalAuthentication&&) &	466
8.18 Header: ara/diag/file_transfer.h	467
8.18.1 Class: FileTransferService	467
8.18.1.1 Public Member Types	467

8.18.1.1.1 Enumeration: WriteFileMode	467
8.18.1.2 Public Member Functions	468
8.18.1.2.1 Special Member Functions	468
8.18.1.2.1.1 Destructor	468
8.18.1.2.2 Constructors	468
8.18.1.2.2.1 FileTransferService	468
8.18.1.2.3 Member Functions	469
8.18.1.2.3.1 DeleteFile	469
8.18.1.2.3.2 Offer	470
8.18.1.2.3.3 RequestReadDirectory	470
8.18.1.2.3.4 RequestReadFile	471
8.18.1.2.3.5 RequestResumeWriteFile	472
8.18.1.2.3.6 RequestWriteFile	473
8.18.1.2.3.7 StopOffer	474
8.18.1.3 Protected Member Functions	474
8.18.1.3.1 Special Member Functions	474
8.18.1.3.1.1 Move Constructor	474
8.18.1.3.2 Constructors	475
8.18.1.3.2.1 FileTransferService	475
8.18.1.3.3 Member Functions	475
8.18.1.3.3.1 operator=(FileTransferService const&)	475
8.18.1.3.3.2 operator=(FileTransferService&&) &	476
8.18.2 Struct: FileSizes	476
8.18.2.1 Public Member Variables	477
8.18.2.1.1 compressed_size	477
8.18.2.1.2 uncompressed_size	477
8.19 Header: ara/diag/generic_data_identifier.h	478
8.19.1 Class: GenericDataIdentifier	478
8.19.1.1 Public Member Functions	478
8.19.1.1.1 Special Member Functions	478
8.19.1.1.1.1 Move Constructor	478
8.19.1.1.1.2 Move Assignment Operator	479
8.19.1.1.1.3 Destructor	479
8.19.1.1.2 Constructors	480
8.19.1.1.2.1 GenericDataIdentifier(const ara::core::Instance Specifier&, DataIdentifierConcurrencyType)	480
8.19.1.1.2.2 GenericDataIdentifier(GenericDataIdentifier&)	480
8.19.1.1.3 Member Functions	481
8.19.1.1.3.1 Offer	481
8.19.1.1.3.2 Read	481
8.19.1.1.3.3 StopOffer	482
8.19.1.1.3.4 Write	483
8.19.1.1.3.5 operator=	483

8.19.2 Struct: OperationOutput	484
8.19.2.1 Public Member Variables	484
8.19.2.1.1 responseData	484
8.20 Header: ara/diag/generic_routine.h	485
8.20.1 Class: GenericRoutine	485
8.20.1.1 Public Member Functions	485
8.20.1.1.1 Special Member Functions	485
8.20.1.1.1.1 Move Constructor	485
8.20.1.1.1.2 Move Assignment Operator	486
8.20.1.1.1.3 Destructor	486
8.20.1.1.2 Constructors	487
8.20.1.1.2.1 GenericRoutine(const ara::core::InstanceSpeci- fier&, ConcurrencyType)	487
8.20.1.1.2.2 GenericRoutine(GenericRoutine&)	487
8.20.1.1.3 Member Functions	488
8.20.1.1.3.1 Offer	488
8.20.1.1.3.2 RequestResults	488
8.20.1.1.3.3 Start	489
8.20.1.1.3.4 Stop	490
8.20.1.1.3.5 StopOffer	491
8.20.1.1.3.6 operator=	491
8.20.2 Struct: OperationOutput	491
8.20.2.1 Public Member Variables	492
8.20.2.1.1 responseData	492
8.21 Header: ara/diag/generic_uds_service.h	492
8.21.1 Class: GenericUDSService	492
8.21.1.1 Public Member Functions	493
8.21.1.1.1 Special Member Functions	493
8.21.1.1.1.1 Move Constructor	493
8.21.1.1.1.2 Move Assignment Operator	493
8.21.1.1.1.3 Destructor	494
8.21.1.1.2 Constructors	494
8.21.1.1.2.1 GenericUDSService(const ara::core::Instance Specifier&, ConcurrencyType)	494
8.21.1.1.2.2 GenericUDSService(GenericUDSService&)	495
8.21.1.1.3 Member Functions	495
8.21.1.1.3.1 HandleMessage	495
8.21.1.1.3.2 Offer	496
8.21.1.1.3.3 StopOffer	497
8.21.1.1.3.4 operator=	497
8.21.2 Struct: OperationOutput	497
8.21.2.1 Public Member Variables	498
8.21.2.1.1 responseData	498

8.22Header: ara/diag/meta_info.h	498
8.22.1 Class: MetaInfo	498
8.22.1.1 Public Member Types	499
8.22.1.1.1 Type Alias: Context	499
8.22.1.2 Public Member Variables	499
8.22.1.2.1 kDiagnosticCommunication	499
8.22.1.2.2 kDoIP	500
8.22.1.2.3 kFaultMemory	500
8.22.1.2.4 kSovd	500
8.22.1.3 Public Member Functions	501
8.22.1.3.1 Special Member Functions	501
8.22.1.3.1.1 Copy Constructor	501
8.22.1.3.1.2 Default Constructor	501
8.22.1.3.1.3 Move Constructor	502
8.22.1.3.1.4 Copy Assignment Operator	502
8.22.1.3.1.5 Move Assignment Operator	502
8.22.1.3.1.6 Destructor	503
8.22.1.3.2 Member Functions	503
8.22.1.3.2.1 GetContext	503
8.22.1.3.2.2 GetValue	504
8.23Header: ara/diag/release_handler.h	504
8.23.1 Class: ReleaseHandler	504
8.23.1.1 Public Member Types	505
8.23.1.1.1 Type Alias: ReleaseHandlerSetNotifier	505
8.23.1.2 Public Member Functions	505
8.23.1.2.1 Special Member Functions	505
8.23.1.2.1.1 Default Constructor	505
8.23.1.2.1.2 Move Constructor	506
8.23.1.2.1.3 Move Assignment Operator	506
8.23.1.2.1.4 Destructor	507
8.23.1.2.2 Constructors	507
8.23.1.2.2.1 ReleaseHandler	507
8.23.1.2.3 Member Functions	508
8.23.1.2.3.1 MayRelease	508
8.23.1.2.3.2 SetNotifier	508
8.23.1.2.3.3 operator=	509
8.24Header: ara/diag/security_access.h	509
8.24.1 Non-Member Types	509
8.24.1.1 Enumeration: KeyCompareResultType	509
8.24.2 Class: SecurityAccess	510
8.24.2.1 Public Member Functions	510
8.24.2.1.1 Special Member Functions	510
8.24.2.1.1.1 Move Constructor	510

8.24.2.1.1.2	Move Assignment Operator	511
8.24.2.1.1.3	Destructor	511
8.24.2.1.2	Constructors	512
8.24.2.1.2.1	SecurityAccess(const ara::core::InstanceSpeci- fier&, ConcurrencyType)	512
8.24.2.1.2.2	SecurityAccess(SecurityAccess&)	512
8.24.2.1.3	Member Functions	513
8.24.2.1.3.1	CompareKey	513
8.24.2.1.3.2	GetSeed	514
8.24.2.1.3.3	Offer	514
8.24.2.1.3.4	StopOffer	515
8.24.2.1.3.5	operator=	515
8.25	Header: ara/diag/service_validation.h	516
8.25.1	Non-Member Types	516
8.25.1.1	Enumeration: ConfirmationStatusType	516
8.25.2	Class: ServiceValidation	517
8.25.2.1	Public Member Functions	517
8.25.2.1.1	Special Member Functions	517
8.25.2.1.1.1	Move Constructor	517
8.25.2.1.1.2	Move Assignment Operator	518
8.25.2.1.1.3	Destructor	518
8.25.2.1.2	Constructors	519
8.25.2.1.2.1	ServiceValidation(const ara::core::InstanceSpecifier&)	519
8.25.2.1.2.2	ServiceValidation(ServiceValidation&)	519
8.25.2.1.3	Member Functions	520
8.25.2.1.3.1	Confirmation	520
8.25.2.1.3.2	Offer	520
8.25.2.1.3.3	StopOffer	521
8.25.2.1.3.4	Validate	521
8.25.2.1.3.5	operator=	522
8.26	Header: ara/diag/transmit_certificate.h	522
8.26.1	Class: TransmitCertificate	522
8.26.1.1	Public Member Functions	523
8.26.1.1.1	Special Member Functions	523
8.26.1.1.1.1	Move Constructor	523
8.26.1.1.1.2	Move Assignment Operator	523
8.26.1.1.1.3	Destructor	524
8.26.1.1.2	Constructors	524
8.26.1.1.2.1	TransmitCertificate(TransmitCertificate&)	524
8.26.1.1.2.2	TransmitCertificate(const ara::core::InstanceSpeci- fier&, ConcurrencyType)	525
8.26.1.1.3	Member Functions	525
8.26.1.1.3.1	Offer	525

8.26.1.1.3.2	Process	526
8.26.1.1.3.3	StopOffer	527
8.26.1.1.3.4	operator=	527
8.27	Header: ara/diag/upload.h	528
8.27.1	Class: UploadService	528
8.27.1.1	Public Member Functions	528
8.27.1.1.1	Special Member Functions	528
8.27.1.1.1.1	Move Constructor	528
8.27.1.1.1.2	Move Assignment Operator	529
8.27.1.1.1.3	Destructor	529
8.27.1.1.2	Constructors	529
8.27.1.1.2.1	UploadService(UploadService&)	529
8.27.1.1.2.2	UploadService(const ara::core::InstanceSpecifier&, ConcurrencyType)	530
8.27.1.1.3	Member Functions	531
8.27.1.1.3.1	Offer	531
8.27.1.1.3.2	RequestUpload	531
8.27.1.1.3.3	RequestUploadExit	532
8.27.1.1.3.4	StopOffer	533
8.27.1.1.3.5	UploadData	533
8.27.1.1.3.6	operator=	534
8.27.2	Struct: OperationOutput	535
8.27.2.1	Public Member Variables	535
8.27.2.1.1	responseData	535
8.28	Header: {<data-element-directory-path>}/{<data-element-shortname-lower>}.h	536
8.28.1	Namespaces	536
8.28.1.1	{<namespace-list-data-element>}	536
8.28.2	Class: {<data-element-interface-name-x>}	537
8.28.2.1	Public Member Variables	537
8.28.2.1.1	{<data-element-out-arg-symbol>}	537
8.28.2.2	Public Member Functions	538
8.28.2.2.1	Member Functions	538
8.28.2.2.1.1	Offer	538
8.28.2.2.1.2	Read	538
8.28.2.2.1.3	StopOffer	539
8.28.2.2.1.4	operator=	540
8.28.2.2.1.5	operator=	540
8.28.2.2.1.6	{<data-element-interface-name-1>}	541
8.28.2.2.1.7	{<data-element-interface-name-2>}	541
8.28.2.2.1.8	{<data-element-interface-name-4>}	542
8.28.2.2.1.9	~{<data-element-interface-name-6>}	542
8.28.3	Struct: {<data-element-name>}ReadOutput	543

8.29 Header: {<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h	544
8.29.1 Namespaces	544
8.29.1.1 {<namespace-list-data-identifier>}	544
8.29.2 Class: {<data-identifier-interface-name-x>}	545
8.29.2.1 Public Member Variables	545
8.29.2.1.1 {<data-identifier-out-arg-symbol>}	545
8.29.2.2 Public Member Functions	546
8.29.2.2.1 Member Functions	546
8.29.2.2.1.1 Offer	546
8.29.2.2.1.2 Read	547
8.29.2.2.1.3 StopOffer	548
8.29.2.2.1.4 Write	548
8.29.2.2.1.5 operator=	550
8.29.2.2.1.6 operator=	551
8.29.2.2.1.7 {<data-identifier-interface-name-1>}	551
8.29.2.2.1.8 {<data-identifier-interface-name-2>}	552
8.29.2.2.1.9 {<data-identifier-interface-name-4>}	552
8.29.2.2.1.10 ~{<data-identifier-interface-name-6>}	553
8.29.3 Struct: {<data-identifier-name>}ReadOutput	553
8.30 Header: {<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h	554
8.30.1 Namespaces	555
8.30.1.1 {<routine-interface-hierarchical-namespace-list>}	555
8.30.2 Class: {<routine-interface-name-x>}	555
8.30.2.1 Public Member Variables	556
8.30.2.1.1 {<routine-interface-request-result-out-arg-symbol>}	556
8.30.2.1.2 {<routine-interface-start-out-arg-symbol>}	556
8.30.2.1.3 {<routine-interface-stop-out-arg-symbol>}	557
8.30.2.2 Public Member Functions	558
8.30.2.2.1 Member Functions	558
8.30.2.2.1.1 Offer	558
8.30.2.2.1.2 RequestResult	558
8.30.2.2.1.3 Start	560
8.30.2.2.1.4 Stop	562
8.30.2.2.1.5 StopOffer	564
8.30.2.2.1.6 operator=	565
8.30.2.2.1.7 operator=	565
8.30.2.2.1.8 {<routine-interface-name-1>}	566
8.30.2.2.1.9 {<routine-interface-name-2>}	566
8.30.2.2.1.10 {<routine-interface-name-4>}	567
8.30.2.2.1.11 ~{<routine-interface-name-6>}	567
8.30.3 Struct: {<routine-interface-name-11>}RequestResultOutput	568

8.30.4 Struct: {<routine-interface-name-6>}StartOutput	569
8.30.5 Struct: {<routine-interface-name-9>}StopOutput	569
8.31 Diagnostic DoIP-related APIs	570
8.32 Header: ara/diag/doip_activationline.h	570
8.32.1 Class: DoIPActivationLine	570
8.32.1.1 Public Member Functions	571
8.32.1.1.1 Special Member Functions	571
8.32.1.1.1.1 Move Constructor	571
8.32.1.1.1.2 Move Assignment Operator	571
8.32.1.1.1.3 Destructor	572
8.32.1.1.2 Constructors	572
8.32.1.1.2.1 DoIPActivationLine(const ara::core::InstanceSpeci- fier&)	572
8.32.1.1.2.2 DoIPActivationLine(DoIPActivationLine&)	573
8.32.1.1.3 Member Functions	573
8.32.1.1.3.1 GetActivationLineState	573
8.32.1.1.3.2 GetNetworkInterfaceId	574
8.32.1.1.3.3 Offer	574
8.32.1.1.3.4 StopOffer	575
8.32.1.1.3.5 UpdateActivationLineState	575
8.32.1.1.3.6 operator=	576
8.33 Header: ara/diag/doip_entity_identification.h	576
8.33.1 Class: DoIPEntityIdentification	576
8.33.1.1 Public Member Functions	577
8.33.1.1.1 Special Member Functions	577
8.33.1.1.1.1 Destructor	577
8.33.1.1.2 Constructors	577
8.33.1.1.2.1 DoIPEntityIdentification	577
8.33.1.1.3 Member Functions	578
8.33.1.1.3.1 GetEntityId	578
8.33.1.1.3.2 Offer	579
8.33.1.1.3.3 StopOffer	579
8.33.2 Struct: EntityId	580
8.33.2.1 Public Member Variables	580
8.33.2.1.1 entityIdentification	580
8.34 Header: ara/diag/doip_group_identification.h	581
8.34.1 Class: DoIPGroupIdentification	581
8.34.1.1 Public Member Functions	581
8.34.1.1.1 Special Member Functions	581
8.34.1.1.1.1 Move Constructor	581
8.34.1.1.1.2 Move Assignment Operator	582
8.34.1.1.1.3 Destructor	582
8.34.1.1.2 Constructors	583

8.34.1.1.2.1	DolPGroupIdentification(const ara::core::InstanceSpecifier&, ConcurrencyType)	583
8.34.1.1.2.2	DolPGroupIdentification(DolPGroupIdentification&)	583
8.34.1.1.3	Member Functions	584
8.34.1.1.3.1	GetGidStatus	584
8.34.1.1.3.2	Offer	584
8.34.1.1.3.3	StopOffer	585
8.34.1.1.3.4	operator=	585
8.34.2	Struct: GidStatus	586
8.35	Header: ara/diag/doip_power_mode.h	586
8.35.1	Non-Member Types	586
8.35.1.1	Enumeration: PowerModeType	586
8.35.2	Class: DolPPowerMode	587
8.35.2.1	Public Member Functions	587
8.35.2.1.1	Special Member Functions	587
8.35.2.1.1.1	Move Constructor	587
8.35.2.1.1.2	Move Assignment Operator	588
8.35.2.1.1.3	Destructor	588
8.35.2.1.2	Constructors	589
8.35.2.1.2.1	DolPPowerMode(const ara::core::InstanceSpecifier&, ConcurrencyType)	589
8.35.2.1.2.2	DolPPowerMode(DolPPowerMode&)	589
8.35.2.1.3	Member Functions	590
8.35.2.1.3.1	GetDolPPowerMode	590
8.35.2.1.3.2	Offer	590
8.35.2.1.3.3	StopOffer	591
8.35.2.1.3.4	operator=	591
8.36	Header: ara/diag/doip_trigger_vehicle_announcement.h	592
8.36.1	Class: DolPTriggerVehicleAnnouncement	592
8.36.1.1	Public Member Functions	592
8.36.1.1.1	Member Functions	592
8.36.1.1.1.1	GetDolPTriggerVehicleAnnouncement	592
8.36.1.1.1.2	TriggerVehicleAnnouncement	593
8.36.1.2	Private Member Functions	593
8.36.1.2.1	Special Member Functions	593
8.36.1.2.1.1	Default Constructor	593
8.36.1.2.1.2	Destructor	594
8.37	Diagnostic Event-related APIs	594
8.38	Header: ara/diag/condition.h	594
8.38.1	Non-Member Types	594
8.38.1.1	Enumeration: ConditionType	594
8.38.2	Class: Condition	595
8.38.2.1	Public Member Functions	595

8.38.2.1.1	Special Member Functions	595
8.38.2.1.1.1	Move Constructor	595
8.38.2.1.1.2	Move Assignment Operator	596
8.38.2.1.1.3	Destructor	596
8.38.2.1.2	Constructors	596
8.38.2.1.2.1	Condition(const ara::core::InstanceSpecifier&)	596
8.38.2.1.2.2	Condition(Condition&)	597
8.38.2.1.3	Member Functions	598
8.38.2.1.3.1	GetCondition	598
8.38.2.1.3.2	SetCondition	598
8.38.2.1.3.3	operator=	599
8.39	Header: ara/diag/operation_cycle.h	599
8.39.1	Class: OperationCycle	599
8.39.1.1	Public Member Types	600
8.39.1.1.1	Type Alias: OperationCycleSetNotifier	600
8.39.1.2	Public Member Functions	600
8.39.1.2.1	Special Member Functions	600
8.39.1.2.1.1	Move Constructor	600
8.39.1.2.1.2	Move Assignment Operator	601
8.39.1.2.1.3	Destructor	601
8.39.1.2.2	Constructors	602
8.39.1.2.2.1	OperationCycle(const ara::core::InstanceSpecifier&)	602
8.39.1.2.2.2	OperationCycle(OperationCycle&)	602
8.39.1.2.3	Member Functions	603
8.39.1.2.3.1	RestartOperationCycle	603
8.39.1.2.3.2	SetNotifier	603
8.39.1.2.3.3	operator=	604
8.40	Header: ara/diag/monitor.h	604
8.40.1	Class: Monitor	604
8.40.1.1	Public Member Functions	605
8.40.1.1.1	Special Member Functions	605
8.40.1.1.1.1	Move Constructor	605
8.40.1.1.1.2	Move Assignment Operator	605
8.40.1.1.2	Constructors	606
8.40.1.1.2.1	Monitor(const ara::core::InstanceSpecifier&, Init MonitorNotifier)	606
8.40.1.1.2.2	Monitor(const ara::core::InstanceSpecifier&, Init MonitorNotifier, CounterBased)	607
8.40.1.1.2.3	Monitor(const ara::core::InstanceSpecifier&, Init MonitorNotifier, TimeBased)	608
8.40.1.1.2.4	Monitor(Monitor&)	609
8.40.1.1.2.5	Monitor(const ara::core::InstanceSpecifier&, Init MonitorNotifier, std::function<std::int8_t(void)>)	609
8.40.1.1.3	Member Functions	610

8.40.1.1.3.1 Offer	610
8.40.1.1.3.2 ReportMonitorAction	610
8.40.1.1.3.3 StopOffer	611
8.40.1.1.3.4 operator=	611
8.41 Header: ara/diag/monitor_types.h	612
8.41.1 Non-Member Types	612
8.41.1.1 Type Alias: InitMonitorNotifier	612
8.41.1.2 Enumeration: InitMonitorReason	612
8.41.1.3 Enumeration: MonitorAction	613
8.41.2 Struct: CounterBased	613
8.41.2.1 Public Member Variables	614
8.41.2.1.1 failedJumpValue	614
8.41.2.1.2 failedStepsize	614
8.41.2.1.3 failedThreshold	615
8.41.2.1.4 passedJumpValue	615
8.41.2.1.5 passedStepsize	616
8.41.2.1.6 passedThreshold	616
8.41.2.1.7 useJumpToFailed	617
8.41.2.1.8 useJumpToPassed	617
8.41.3 Struct: TimeBased	617
8.41.3.1 Public Member Variables	618
8.41.3.1.1 failedMs	618
8.41.3.1.2 passedMs	618
8.42 Header: ara/diag/multiple_condition.h	619
8.42.1 Non-Member Types	619
8.42.1.1 Type Alias: ConditionHandleType	619
8.42.2 Class: MultipleCondition	619
8.42.2.1 Public Member Functions	620
8.42.2.1.1 Special Member Functions	620
8.42.2.1.1.1 Move Constructor	620
8.42.2.1.1.2 Move Assignment Operator	620
8.42.2.1.1.3 Destructor	621
8.42.2.1.2 Constructors	621
8.42.2.1.2.1 MultipleCondition(const ara::core::InstanceSpecifier&)	621
8.42.2.1.2.2 MultipleCondition(MultipleCondition&)	622
8.42.2.1.3 Member Functions	622
8.42.2.1.3.1 GetCondition	622
8.42.2.1.3.2 SetCondition	623
8.42.2.1.3.3 operator=	623
8.43 Header: ara/diag/multiple_event.h	624
8.43.1 Non-Member Types	624
8.43.1.1 Type Alias: EventHandleType	624
8.43.2 Class: MultipleEvent	624

8.43.2.1 Public Member Functions	625
8.43.2.1.1 Special Member Functions	625
8.43.2.1.1.1 Move Constructor	625
8.43.2.1.1.2 Move Assignment Operator	625
8.43.2.1.1.3 Destructor	625
8.43.2.1.2 Constructors	626
8.43.2.1.2.1 MultipleEvent(MultipleEvent&)	626
8.43.2.1.2.2 MultipleEvent(const ara::core::InstanceSpecifier&)	626
8.43.2.1.3 Member Functions	627
8.43.2.1.3.1 GetDTCNumber	627
8.43.2.1.3.2 GetDebouncingStatus	628
8.43.2.1.3.3 GetEventStatus	628
8.43.2.1.3.4 GetFaultDetectionCounter	629
8.43.2.1.3.5 SetEventStatusChangedNotifier	630
8.43.2.1.3.6 operator=	630
8.44 Header: ara/diag/multiple_monitor.h	631
8.44.1 Non-Member Types	631
8.44.1.1 Type Alias: MonitorHandleType	631
8.44.2 Class: MultipleMonitor	631
8.44.2.1 Public Member Functions	632
8.44.2.1.1 Special Member Functions	632
8.44.2.1.1.1 Move Constructor	632
8.44.2.1.1.2 Move Assignment Operator	632
8.44.2.1.2 Constructors	633
8.44.2.1.2.1 MultipleMonitor	633
8.44.2.1.2.2 MultipleMonitor	633
8.44.2.1.3 Member Functions	634
8.44.2.1.3.1 ConfigureMonitor(MonitorHandleType, InitMonitor Notifier, CounterBased)	634
8.44.2.1.3.2 ConfigureMonitor(MonitorHandleType, InitMonitor Notifier, TimeBased)	634
8.44.2.1.3.3 ConfigureMonitor(MonitorHandleType, InitMonitor Notifier)	635
8.44.2.1.3.4 ConfigureMonitor(MonitorHandleType, InitMonitor Notifier, std::function<std::int8_t(void)>)	636
8.44.2.1.3.5 Offer	636
8.44.2.1.3.6 ReportMonitorAction	637
8.44.2.1.3.7 StopOffer	637
8.44.2.1.3.8 operator=	638
8.45 Header: ara/diag/indicator.h	638
8.45.1 Non-Member Types	638
8.45.1.1 Enumeration: IndicatorStatusType	638
8.45.2 Class: Indicator	639
8.45.2.1 Public Member Types	639

8.45.2.1.1 Type Alias: IndicatorStatusTypeNotifier	639
8.45.2.2 Public Member Functions	640
8.45.2.2.1 Special Member Functions	640
8.45.2.2.1.1 Move Constructor	640
8.45.2.2.1.2 Move Assignment Operator	640
8.45.2.2.1.3 Destructor	640
8.45.2.2.2 Constructors	641
8.45.2.2.2.1 Indicator(const ara::core::InstanceSpecifier&)	641
8.45.2.2.2.2 Indicator(Indicator&)	642
8.45.2.2.3 Member Functions	642
8.45.2.2.3.1 GetIndicatorState	642
8.45.2.2.3.2 SetNotifier	643
8.45.2.2.3.3 operator=	643
8.46Header: ara/diag/event.h	644
8.46.1 Class: Event	644
8.46.1.1 Public Member Functions	644
8.46.1.1.1 Special Member Functions	644
8.46.1.1.1.1 Move Constructor	644
8.46.1.1.1.2 Move Assignment Operator	645
8.46.1.1.1.3 Destructor	645
8.46.1.1.2 Constructors	645
8.46.1.1.2.1 Event(const ara::core::InstanceSpecifier&)	645
8.46.1.1.2.2 Event(Event&)	646
8.46.1.1.3 Member Functions	647
8.46.1.1.3.1 GetDTCNumber	647
8.46.1.1.3.2 GetDebouncingStatus	647
8.46.1.1.3.3 GetEventStatus	648
8.46.1.1.3.4 GetFaultDetectionCounter	648
8.46.1.1.3.5 GetTestComplete	649
8.46.1.1.3.6 SetEventStatusChangedNotifier	649
8.46.1.1.3.7 operator=	650
8.47Header: ara/diag/event_types.h	650
8.47.1 Non-Member Types	650
8.47.1.1 Enumeration: DTCFormatType	650
8.47.1.2 Enumeration: DebouncingState	651
8.47.1.3 Enumeration: EventStatusBit	651
8.47.1.4 Type Alias: EventStatusByteNotifier	652
8.47.2 Struct: EventStatusByte	652
8.47.2.1 Public Member Functions	653
8.47.2.1.1 Special Member Functions	653
8.47.2.1.1.1 Copy Constructor	653
8.47.2.1.1.2 Move Constructor	653
8.47.2.1.1.3 Copy Assignment Operator	654

8.47.2.1.1.4 Move Assignment Operator	654
8.47.2.1.2 Constructors	655
8.47.2.1.2.1 EventStatusByte	655
8.47.2.1.3 Member Functions	655
8.47.2.1.3.1 IsFailedAndTested	655
8.47.2.1.3.2 IsNotSet	656
8.47.2.1.3.3 IsPassedAndTested	656
8.47.2.1.3.4 IsSet	657
8.48 Header: ara/diag/dtc_information.h	657
8.48.1 Non-Member Types	657
8.48.1.1 Enumeration: ControlDtcStatusType	657
8.48.1.2 Enumeration: UdsDtcStatusBitType	658
8.48.2 Class: DTCInformation	658
8.48.2.1 Public Member Types	659
8.48.2.1.1 Type Alias: ControlDtcStatusTypeNotifier	659
8.48.2.1.2 Type Alias: DtcStatusChangedNotifier	659
8.48.2.1.3 Type Alias: EventMemoryOverflowSetNotifier	660
8.48.2.1.4 Type Alias: NumberOfStoredEntriesNotifier	660
8.48.2.1.5 Type Alias: SnapshotRecordUpdatedTypeNotifier	661
8.48.2.2 Public Member Functions	661
8.48.2.2.1 Special Member Functions	661
8.48.2.2.1.1 Move Constructor	661
8.48.2.2.1.2 Move Assignment Operator	662
8.48.2.2.1.3 Destructor	662
8.48.2.2.2 Constructors	662
8.48.2.2.2.1 DTCInformation(const ara::core::InstanceSpecifier&)	662
8.48.2.2.2.2 DTCInformation(DTCInformation&)	663
8.48.2.2.3 Member Functions	664
8.48.2.2.3.1 Clear	664
8.48.2.2.3.2 EnableControlDtc	664
8.48.2.2.3.3 GetControlDTCStatus	665
8.48.2.2.3.4 GetCurrentStatus	665
8.48.2.2.3.5 GetEventMemoryOverflow	666
8.48.2.2.3.6 GetNumberOfStoredEntries	667
8.48.2.2.3.7 SetControlDtcStatusNotifier	667
8.48.2.2.3.8 SetDTCStatusChangedNotifier	668
8.48.2.2.3.9 SetEventMemoryOverflowNotifier	668
8.48.2.2.3.10 SetNumberOfStoredEntriesNotifier	669
8.48.2.2.3.11 SetSnapshotRecordUpdatedNotifier	669
8.48.2.2.3.12 operator=	670
8.48.3 Struct: SnapshotDataIdentifierType	670
8.48.3.1 Public Member Variables	671
8.48.3.1.1 data	671

8.48.3.1.2 dataIdentifier	671
8.48.4 Struct: SnapshotDataRecordType	672
8.48.4.1 Public Member Variables	672
8.48.4.1.1 snapshotDataIdentifiers	672
8.48.4.1.2 snapshotRecordNumber	673
8.48.5 Struct: SnapshotRecordUpdatedType	673
8.48.5.1 Public Member Variables	674
8.48.5.1.1 DTC	674
8.48.5.1.2 snapshotDataRecords	674
8.48.6 Struct: UdsDtcStatusByteType	675
8.48.6.1 Public Member Functions	675
8.48.6.1.1 Special Member Functions	675
8.48.6.1.1.1 Move Constructor	675
8.48.6.1.1.2 Copy Constructor	676
8.48.6.1.1.3 Copy Assignment Operator	676
8.48.6.1.1.4 Move Assignment Operator	677
8.48.6.1.2 Constructors	677
8.48.6.1.2.1 UdsDtcStatusByteType	677
8.48.6.1.3 Member Functions	678
8.48.6.1.3.1 IsFailedAndTested	678
8.48.6.1.3.2 IsNotSet	678
8.48.6.1.3.3 IsPassedAndTested	679
8.48.6.1.3.4 IsSet	679
8.49 Header: {<extended-data-record-element-shortname-lower>}.h	680
8.49.1 Namespaces	680
8.49.1.1 {<namespace-list-extended-data-record-element>}	680
8.49.2 Class: {<extended-data-record-element-interface-name-x>}	681
8.49.2.1 Public Member Variables	681
8.49.2.1.1 {<extended-data-record-element-out-arg-symbol>}	681
8.49.2.2 Public Member Functions	682
8.49.2.2.1 Member Functions	682
8.49.2.2.1.1 Offer	682
8.49.2.2.1.2 Read	683
8.49.2.2.1.3 StopOffer	684
8.49.2.2.1.4 operator=	684
8.49.2.2.1.5 operator=	685
8.49.2.2.1.6 {<extended-data-record-element-interface-name-1>}	685
8.49.2.2.1.7 {<extended-data-record-element-interface-name-2>}	686
8.49.2.2.1.8 {<extended-data-record-element-interface-name-4>}	686
8.49.2.2.1.9 ~{<extended-data-record-element-interface-name-6>}	687
8.49.3 Struct: {<extended-data-record-element-name-upper-camel>}ReadOutput	688
8.50 Diagnostic SOVD-related APIs	688

8.51 Header: ara/diag/diag_sovd_error_domain.h	688
8.51.1 Non-Member Types	688
8.51.1.1 Enumeration: DiagSovdErrc	688
8.51.2 Non-Member Functions	689
8.51.2.1 Other	689
8.51.2.1.1 GetDiagSovdErrorDomain	689
8.51.2.1.2 MakeErrorCode	690
8.51.3 Class: DiagSovdErrorDomain	690
8.51.3.1 Public Member Types	691
8.51.3.1.1 Type Alias: Errc	691
8.51.3.1.2 Type Alias: Exception	691
8.51.3.2 Public Member Functions	692
8.51.3.2.1 Special Member Functions	692
8.51.3.2.1.1 Default Constructor	692
8.51.3.2.2 Member Functions	692
8.51.3.2.2.1 Message	692
8.51.3.2.2.2 Name	693
8.51.3.2.2.3 ThrowAsException	693
8.51.4 Class: DiagSovdException	694
8.51.4.1 Public Member Functions	694
8.51.4.1.1 Constructors	694
8.51.4.1.1.1 DiagSovdException	694
8.52 Header: ara/diag/sovd_authorization.h	695
8.52.1 Class: SovdAuthorization	695
8.52.1.1 Public Member Functions	695
8.52.1.1.1 Special Member Functions	695
8.52.1.1.1.1 Destructor	695
8.52.1.1.2 Constructors	696
8.52.1.1.2.1 SovdAuthorization	696
8.52.1.1.3 Member Functions	696
8.52.1.1.3.1 GetAuthorizationUrl	696
8.52.1.1.3.2 GetTokenUrl	697
8.52.1.1.3.3 Offer	698
8.52.1.1.3.4 StopOffer	698
8.52.1.1.3.5 ValidateAuthorization	699
8.52.2 Struct: ValidateAuthorizationOutput	700
8.52.2.1 Public Member Variables	700
8.52.2.1.1 identity	700
8.52.2.1.2 validUntil	701
8.53 Header: ara/diag/sovd_bulk_data.h	701
8.53.1 Class: SovdBulkData	701
8.53.1.1 Public Member Functions	702
8.53.1.1.1 Special Member Functions	702

8.53.1.1.1.1 Destructor	702
8.53.1.1.2 Constructors	702
8.53.1.1.2.1 SovdBulkData	702
8.53.1.1.3 Member Functions	703
8.53.1.1.3.1 DeleteAllBulkData	703
8.53.1.1.3.2 DeleteSpecificBulkData	704
8.53.1.1.3.3 GetBulkDataMetaData	704
8.53.1.1.3.4 Offer	705
8.53.1.1.3.5 RequestBulkDataDownload	705
8.53.1.1.3.6 RequestBulkDataUpload	706
8.53.1.1.3.7 StopOffer	707
8.53.1.2 Protected Member Functions	707
8.53.1.2.1 Special Member Functions	707
8.53.1.2.1.1 Move Constructor	707
8.53.1.2.1.2 Move Assignment Operator	708
8.53.1.2.2 Constructors	708
8.53.1.2.2.1 SovdBulkData	708
8.53.1.2.3 Member Functions	709
8.53.1.2.3.1 operator=	709
8.53.2 Struct: BulkDataMetaDataDescriptor	709
8.53.2.1 Public Member Variables	710
8.53.2.1.1 creation_date	710
8.53.2.1.2 file_name	710
8.53.2.1.3 hash	711
8.53.2.1.4 hash_algorithm	711
8.53.2.1.5 id	712
8.53.2.1.6 last_modified	712
8.53.2.1.7 mimetype	713
8.53.2.1.8 name	713
8.53.2.1.9 size	714
8.53.2.1.10 translation_id	714
8.53.3 Struct: DeleteByCategoryResult	715
8.53.3.1 Public Member Variables	715
8.53.3.1.1 deleted_items	715
8.53.3.1.2 failed_items	716
8.53.4 Struct: DeletionError	716
8.53.4.1 Public Member Variables	717
8.53.4.1.1 error	717
8.53.4.1.2 item	717
8.54 Header: ara/diag/sovd_configuration.h	718
8.54.1 Class: SovdConfiguration	718
8.54.1.1 Public Member Functions	718
8.54.1.1.1 Special Member Functions	718

8.54.1.1.1.1 Destructor	718
8.54.1.1.2 Constructors	719
8.54.1.1.2.1 SovdConfiguration	719
8.54.1.1.3 Member Functions	719
8.54.1.1.3.1 Offer	719
8.54.1.1.3.2 RequestGetConfiguration	720
8.54.1.1.3.3 RequestPutConfiguration	721
8.54.1.1.3.4 StopOffer	721
8.54.1.2 Protected Member Functions	722
8.54.1.2.1 Special Member Functions	722
8.54.1.2.1.1 Move Constructor	722
8.54.1.2.1.2 Move Assignment Operator	722
8.54.1.2.2 Constructors	723
8.54.1.2.2.1 SovdConfiguration	723
8.54.1.2.3 Member Functions	723
8.54.1.2.3.1 operator=	723
8.54.2 Struct: SovdConfigurationMetaInfo	724
8.54.2.1 Public Member Variables	724
8.54.2.1.1 mimetype	724
8.54.2.1.2 size	725
8.55Header: ara/diag/sovd_http_redirect.h	725
8.55.1 Struct: HttpRedirect	725
8.55.1.1 Public Member Variables	726
8.55.1.1.1 url	726
8.56Header: ara/diag/sovd_proximity_challenge.h	726
8.56.1 Non-Member Types	726
8.56.1.1 Enumeration: ProximityChallengeResultType	726
8.56.1.2 Type Alias: ProximityChallengeValidationResultType	727
8.56.2 Class: SovdProximityChallenge	727
8.56.2.1 Public Member Functions	728
8.56.2.1.1 Special Member Functions	728
8.56.2.1.1.1 Destructor	728
8.56.2.1.2 Constructors	728
8.56.2.1.2.1 SovdProximityChallenge	728
8.56.2.1.3 Member Functions	729
8.56.2.1.3.1 GetChallenge	729
8.56.2.1.3.2 Offer	730
8.56.2.1.3.3 StopOffer	730
8.56.2.1.3.4 ValidateResponse	731
8.56.3 Struct: SovdProximityChallengeType	731
8.56.3.1 Public Member Variables	732
8.56.3.1.1 challenge	732
8.56.3.1.2 valid_until	732

8.57 Header: ara/diag/sovd_service_validation.h	733
8.57.1 Non-Member Types	733
8.57.1.1 Enumeration: SovdRequestMethod	733
8.57.2 Class: SovdServiceValidation	733
8.57.2.1 Public Member Functions	734
8.57.2.1.1 Special Member Functions	734
8.57.2.1.1.1 Destructor	734
8.57.2.1.2 Constructors	734
8.57.2.1.2.1 SovdServiceValidation	734
8.57.2.1.3 Member Functions	735
8.57.2.1.3.1 Offer	735
8.57.2.1.3.2 StopOffer	736
8.57.2.1.3.3 Validate	736
8.58 Header: ara/diag/sovd_sw_update.h	737
8.58.1 Non-Member Types	737
8.58.1.1 Enumeration: SovdUpdatePhase	737
8.58.1.2 Enumeration: SovdUpdateStatus	738
8.58.2 Class: SovdSwUpdate	738
8.58.2.1 Public Member Functions	739
8.58.2.1.1 Special Member Functions	739
8.58.2.1.1.1 Destructor	739
8.58.2.1.2 Constructors	739
8.58.2.1.2.1 SovdSwUpdate	739
8.58.2.1.3 Member Functions	740
8.58.2.1.3.1 DeleteUpdatePackage	740
8.58.2.1.3.2 ExecuteUpdatePackage	741
8.58.2.1.3.3 GetAllUpdates	741
8.58.2.1.3.4 GetUpdatePackageDetails	742
8.58.2.1.3.5 GetUpdatePackageStatus	743
8.58.2.1.3.6 Offer	743
8.58.2.1.3.7 PrepareUpdatePackage	744
8.58.2.1.3.8 PutUpdatePackageAutomated	744
8.58.2.1.3.9 RequestUpdatePackageRegistration	745
8.58.2.1.3.10 StopOffer	746
8.58.2.2 Protected Member Functions	746
8.58.2.2.1 Special Member Functions	746
8.58.2.2.1.1 Move Constructor	746
8.58.2.2.1.2 Move Assignment Operator	747
8.58.2.2.2 Constructors	747
8.58.2.2.2.1 SovdSwUpdate	747
8.58.2.2.3 Member Functions	748
8.58.2.2.3.1 operator=	748
8.58.3 Struct: SubProgress	748

8.58.3.1 Public Member Variables	749
8.58.3.1.1 entity	749
8.58.3.1.2 error	749
8.58.3.1.3 progress	750
8.58.3.1.4 status	750
8.58.4 Struct: UpdatePackageDetails	751
8.58.4.1 Public Member Variables	751
8.58.4.1.1 affected_components	751
8.58.4.1.2 automated	752
8.58.4.1.3 duration	752
8.58.4.1.4 execution_conditions	753
8.58.4.1.5 id	753
8.58.4.1.6 name	754
8.58.4.1.7 notes	754
8.58.4.1.8 notes_translation_id	755
8.58.4.1.9 origin	755
8.58.4.1.10 preconditions	756
8.58.4.1.11 preconditions_translation_id	756
8.58.4.1.12 size	757
8.58.4.1.13 translation_id	757
8.58.4.1.14 updated_components	758
8.58.4.1.15 user_activity	758
8.58.4.1.16 user_activity_translation_id	759
8.58.5 Struct: UpdatePackageStatus	759
8.58.5.1 Public Member Variables	760
8.58.5.1.1 error	760
8.58.5.1.2 phase	760
8.58.5.1.3 progress	761
8.58.5.1.4 status	761
8.58.5.1.5 step	762
8.58.5.1.6 step_translation_id	762
8.58.5.1.7 subprogress	763
8.59 Header: ara/diag/diagnostic_sovd_content_data.h	763
8.59.1 Struct: DiagnosticSovdContentData	763
8.59.1.1 Public Member Variables	764
8.59.1.1.1 elements	764
8.60 Header: ara/diag/diagnostic_sovd_content_element_data.h	764
8.60.1 Non-Member Types	764
8.60.1.1 Enumeration: DiagnosticSovdContentElementDataType	764
8.60.1.2 Enumeration: DiagnosticSovdPrimitiveContentValueType	765
8.60.2 Class: DiagnosticSovdArrayContentElementDataConstIterator	765
8.60.2.1 Public Member Functions	766
8.60.2.1.1 Member Functions	766

8.60.2.1.1.1	operator!=	766
8.60.2.1.1.2	operator*	766
8.60.2.1.1.3	operator++	767
8.60.2.1.1.4	operator->	767
8.60.2.1.1.5	operator==	768
8.60.3	Class: DiagnosticSovdArrayContentElementDataConstView	768
8.60.3.1	Public Member Functions	769
8.60.3.1.1	Member Functions	769
8.60.3.1.1.1	Begin	769
8.60.3.1.1.2	CBegin	769
8.60.3.1.1.3	CEnd	770
8.60.3.1.1.4	Capacity	770
8.60.3.1.1.5	Empty	771
8.60.3.1.1.6	End	771
8.60.3.1.1.7	HasError	772
8.60.3.1.1.8	Size	772
8.60.3.1.1.9	operator[]	773
8.60.4	Class: DiagnosticSovdArrayContentElementDataIterator	773
8.60.4.1	Public Member Functions	774
8.60.4.1.1	Member Functions	774
8.60.4.1.1.1	operator!=	774
8.60.4.1.1.2	operator*	774
8.60.4.1.1.3	operator++	775
8.60.4.1.1.4	operator->	775
8.60.4.1.1.5	operator==	776
8.60.5	Class: DiagnosticSovdArrayContentElementDataView	776
8.60.5.1	Public Member Functions	777
8.60.5.1.1	Member Functions	777
8.60.5.1.1.1	Begin	777
8.60.5.1.1.2	CBegin	777
8.60.5.1.1.3	CEnd	778
8.60.5.1.1.4	Capacity	778
8.60.5.1.1.5	Clear	779
8.60.5.1.1.6	Empty	779
8.60.5.1.1.7	End	780
8.60.5.1.1.8	Erase	780
8.60.5.1.1.9	HasError	781
8.60.5.1.1.10	PushBack	781
8.60.5.1.1.11	PushBack	782
8.60.5.1.1.12	PushBack	782
8.60.5.1.1.13	PushBack	783
8.60.5.1.1.14	PushBack	783
8.60.5.1.1.15	PushBack	784

8.60.5.1.1.16 Reserve	784
8.60.5.1.1.17 SetError	785
8.60.5.1.1.18 Size	785
8.60.5.1.1.19 operator[]	786
8.60.5.1.1.20 operator[]	786
8.60.6 Class: DiagnosticSovdContentElementData	787
8.60.6.1 Public Member Functions	787
8.60.6.1.1 Special Member Functions	787
8.60.6.1.1.1 Destructor	787
8.60.6.1.2 Member Functions	788
8.60.6.1.2.1 Add	788
8.60.6.1.2.2 Add	788
8.60.6.1.2.3 Add	789
8.60.6.1.2.4 Add	789
8.60.6.1.2.5 Add	790
8.60.6.1.2.6 Add	791
8.60.6.1.2.7 Begin	791
8.60.6.1.2.8 CBegin	792
8.60.6.1.2.9 CEnd	792
8.60.6.1.2.10 Capacity	793
8.60.6.1.2.11 Clear	793
8.60.6.1.2.12 CreateArrayContentElement	794
8.60.6.1.2.13 CreateContentElement	794
8.60.6.1.2.14 CreatePrimitiveContentBool	795
8.60.6.1.2.15 CreatePrimitiveContentDouble	795
8.60.6.1.2.16 CreatePrimitiveContentSignedInt	796
8.60.6.1.2.17 CreatePrimitiveContentString	796
8.60.6.1.2.18 CreatePrimitiveContentUnsignedInt	797
8.60.6.1.2.19 CreateRecordContentElement	797
8.60.6.1.2.20 Empty	798
8.60.6.1.2.21 End	798
8.60.6.1.2.22 Erase	799
8.60.6.1.2.23 Erase	799
8.60.6.1.2.24 Find	800
8.60.6.1.2.25 Find	800
8.60.6.1.2.26 GetArrayView	801
8.60.6.1.2.27 GetArrayView	801
8.60.6.1.2.28 GetBool	802
8.60.6.1.2.29 GetDouble	802
8.60.6.1.2.30 GetPrimitiveContentValueType	803
8.60.6.1.2.31 GetRecordView	803
8.60.6.1.2.32 GetRecordView	804
8.60.6.1.2.33 GetSignedInt	804

8.60.6.1.2.34 GetString	805
8.60.6.1.2.35 GetType	805
8.60.6.1.2.36 GetUnsignedInt	806
8.60.6.1.2.37 HasContentElement	806
8.60.6.1.2.38 HasError	807
8.60.6.1.2.39 PushBack	807
8.60.6.1.2.40 PushBack	808
8.60.6.1.2.41 PushBack	808
8.60.6.1.2.42 PushBack	809
8.60.6.1.2.43 PushBack	809
8.60.6.1.2.44 PushBack	810
8.60.6.1.2.45 RecordBegin	810
8.60.6.1.2.46 RecordCBegin	811
8.60.6.1.2.47 RecordCEnd	811
8.60.6.1.2.48 RecordCapacity	812
8.60.6.1.2.49 RecordCount	812
8.60.6.1.2.50 RecordEmpty	813
8.60.6.1.2.51 RecordEnd	813
8.60.6.1.2.52 RecordRemoveAll	814
8.60.6.1.2.53 RecordReserve	814
8.60.6.1.2.54 Remove	815
8.60.6.1.2.55 Reserve	816
8.60.6.1.2.56 SetBool	816
8.60.6.1.2.57 SetDouble	817
8.60.6.1.2.58 SetError	817
8.60.6.1.2.59 SetSignedInt	818
8.60.6.1.2.60 SetString	818
8.60.6.1.2.61 SetUnsignedInt	819
8.60.6.1.2.62 Size	819
8.60.6.1.2.63 operator[]	820
8.60.6.1.2.64 operator[]	820
8.60.6.1.2.65 operator[]	821
8.60.6.1.2.66 operator[]	821
8.60.7 Class: DiagnosticSovdContentElementDataWithKey	822
8.60.7.1 Public Member Functions	822
8.60.7.1.1 Special Member Functions	822
8.60.7.1.1.1 Destructor	822
8.60.7.1.2 Member Functions	823
8.60.7.1.2.1 GetContent	823
8.60.7.1.2.2 GetContent	823
8.60.7.1.2.3 GetName	824
8.60.7.1.2.4 SetContent	824
8.60.7.1.2.5 SetName	825

8.60.8 Class: DiagnosticSovdRecordContentElementDataConstIterator	825
8.60.8.1 Public Member Functions	826
8.60.8.1.1 Member Functions	826
8.60.8.1.1.1 operator!=	826
8.60.8.1.1.2 operator*	826
8.60.8.1.1.3 operator++	827
8.60.8.1.1.4 operator->	827
8.60.8.1.1.5 operator==	828
8.60.9 Class: DiagnosticSovdRecordContentElementDataConstView	828
8.60.9.1 Public Member Functions	829
8.60.9.1.1 Member Functions	829
8.60.9.1.1.1 Begin	829
8.60.9.1.1.2 CBegin	829
8.60.9.1.1.3 CEnd	830
8.60.9.1.1.4 Capacity	830
8.60.9.1.1.5 Empty	831
8.60.9.1.1.6 End	831
8.60.9.1.1.7 Find	832
8.60.9.1.1.8 HasContentElement	832
8.60.9.1.1.9 HasError	833
8.60.9.1.1.10 Size	833
8.60.9.1.1.11 operator[]	834
8.60.10 Class: DiagnosticSovdRecordContentElementDataIterator	834
8.60.10.1 Public Member Functions	835
8.60.10.1.1 Member Functions	835
8.60.10.1.1.1 operator!=	835
8.60.10.1.1.2 operator*	835
8.60.10.1.1.3 operator++	836
8.60.10.1.1.4 operator->	836
8.60.10.1.1.5 operator==	837
8.60.11 Class: DiagnosticSovdRecordContentElementDataView	837
8.60.11.1 Public Member Functions	838
8.60.11.1.1 Member Functions	838
8.60.11.1.1.1 Add	838
8.60.11.1.1.2 Add	838
8.60.11.1.1.3 Add	839
8.60.11.1.1.4 Add	839
8.60.11.1.1.5 Add	840
8.60.11.1.1.6 Add	841
8.60.11.1.1.7 Begin	841
8.60.11.1.1.8 CBegin	842
8.60.11.1.1.9 CEnd	842
8.60.11.1.1.10 Capacity	843

8.60.11.1.1.11 Clear	843
8.60.11.1.1.12 Empty	844
8.60.11.1.1.13 End	844
8.60.11.1.1.14 Erase	845
8.60.11.1.1.15 Find	845
8.60.11.1.1.16 Find	846
8.60.11.1.1.17 HasContentElement	846
8.60.11.1.1.18 HasError	847
8.60.11.1.1.19 Remove	847
8.60.11.1.1.20 Reserve	848
8.60.11.1.1.21 SetError	848
8.60.11.1.1.22 Size	849
8.60.11.1.1.23 operator[]	849
8.60.11.1.1.24 operator[]	850
8.61 Header: ara/diag/generic_diagnostic_sovd_content.h	850
8.61.1 Class: GenericDiagnosticSovdContent	850
8.61.1.1 Public Member Functions	851
8.61.1.1.1 Special Member Functions	851
8.61.1.1.1.1 Destructor	851
8.61.1.1.2 Constructors	851
8.61.1.1.2.1 GenericDiagnosticSovdContent	851
8.61.1.1.3 Member Functions	852
8.61.1.1.3.1 Offer	852
8.61.1.1.3.2 Read	852
8.61.1.1.3.3 StopOffer	853
8.61.1.1.3.4 Write	853
8.61.1.2 Protected Member Functions	854
8.61.1.2.1 Special Member Functions	854
8.61.1.2.1.1 Move Constructor	854
8.61.1.2.2 Member Functions	854
8.61.1.2.2.1 operator=	854
8.62 Header: ara/diag/generic_diagnostic_sovd_operation.h	855
8.62.1 Non-Member Types	855
8.62.1.1 Enumeration: OperationExecutionCapability	855
8.62.1.2 Enumeration: OperationExecutionStatus	855
8.62.1.3 Enumeration: TerminateOperationExecutionReason	856
8.62.2 Class: GenericDiagnosticSovdOperation	856
8.62.2.1 Public Member Functions	857
8.62.2.1.1 Special Member Functions	857
8.62.2.1.1.1 Destructor	857
8.62.2.1.2 Constructors	857
8.62.2.1.2.1 GenericDiagnosticSovdOperation	857
8.62.2.1.3 Member Functions	858

8.62.2.1.3.1 Offer	858
8.62.2.1.3.2 StartExecution	858
8.62.2.1.3.3 StopOffer	859
8.62.2.2 Protected Member Functions	860
8.62.2.2.1 Special Member Functions	860
8.62.2.2.1.1 Move Constructor	860
8.62.2.2.2 Member Functions	860
8.62.2.2.2.1 operator=	860
8.62.3 Struct: StartAsynchronousExecutionResponse	861
8.62.4 Struct: StartSynchronousExecutionResponse	861
8.62.5 Class: SovdOperationExecution	861
8.62.5.1 Public Member Functions	862
8.62.5.1.1 Special Member Functions	862
8.62.5.1.1.1 Default Constructor	862
8.62.5.1.1.2 Destructor	862
8.62.5.2 Protected Member Functions	863
8.62.5.2.1 Member Functions	863
8.62.5.2.1.1 ExecuteCapability	863
8.62.5.2.1.2 GetStatus	863
8.62.5.2.1.3 Terminate	864
8.62.6 Struct: GetStatusResponse	865
9 Service Interfaces	866
10 Configuration	867
10.1 Default Values	867
10.2 Semantic Constraints	867
A Mentioned Manifest Elements	869
B Demands and constraints on Base Software (normative)	952
C Platform Extension Interfaces (normative)	953
C.1 Header: apext/diag/uds_transport/protocol_handler.h	953
C.1.1 Class: UdsTransportProtocolHandler	953
C.1.1.1 Public Member Types	953
C.1.1.1.1 Enumeration: InitializationResult	953
C.1.1.2 Protected Member Variables	954
C.1.1.2.1 transportprotocolManager	954
C.1.1.3 Public Member Functions	954
C.1.1.3.1 Special Member Functions	954
C.1.1.3.1.1 Destructor	954
C.1.1.3.2 Constructors	955
C.1.1.3.2.1 UdsTransportProtocolHandler	955
C.1.1.3.3 Member Functions	955
C.1.1.3.3.1 GetHandlerID	955

C.1.1.3.3.2	GetPeriodicHandler	956
C.1.1.3.3.3	Initialize	956
C.1.1.3.3.4	NotifyReestablishment	957
C.1.1.3.3.5	Start	958
C.1.1.3.3.6	Stop	959
C.1.1.3.3.7	Transmit	959
C.2	Header: apex/diag/uds_transport/protocol_mgr.h	960
C.2.1	Class: UdsTransportProtocolMgr	960
C.2.1.1	Public Member Types	960
C.2.1.1.1	Type Alias: GlobalChannelIdentifier	960
C.2.1.1.2	Enumeration: IndicationResult	961
C.2.1.1.3	Enumeration: TransmissionResult	961
C.2.1.2	Public Member Functions	962
C.2.1.2.1	Special Member Functions	962
C.2.1.2.1.1	Destructor	962
C.2.1.2.2	Member Functions	962
C.2.1.2.2.1	ChannelReestablished	962
C.2.1.2.2.2	HandleMessage	963
C.2.1.2.2.3	HandlerStopped	963
C.2.1.2.2.4	IndicateMessage	964
C.2.1.2.2.5	NotifyMessageFailure	965
C.2.1.2.2.6	PeriodicTransmitConfirmation	965
C.2.1.2.2.7	TransmitConfirmation	966
C.3	Header: apex/diag/uds_transport/protocol_periodic_handler.h	966
C.3.1	Class: UdsTransportProtocolPeriodicHandler	966
C.3.1.1	Public Member Functions	967
C.3.1.1.1	Special Member Functions	967
C.3.1.1.1.1	Destructor	967
C.3.1.1.2	Member Functions	967
C.3.1.1.2.1	GetMaxPayloadLength	967
C.3.1.1.2.2	GetNumberOfPeriodicMessages	968
C.3.1.1.2.3	PeriodicTransmit	968
C.4	Header: apex/diag/uds_transport/protocol_types.h	969
C.4.1	Non-Member Types	969
C.4.1.1	Type Alias: ByteSpan	969
C.4.1.2	Type Alias: ChannelID	969
C.4.1.3	Type Alias: UdsTransportProtocolHandlerID	970
C.5	Header: apex/diag/uds_transport/uds_message.h	970
C.5.1	Non-Member Types	970
C.5.1.1	Type Alias: UdsMessageConstPtr	970
C.5.1.2	Type Alias: UdsMessagePtr	971
C.5.2	Class: UdsMessage	971
C.5.2.1	Public Member Types	972

C.5.2.1.1	Type Alias: Address	972
C.5.2.1.2	Type Alias: MetaInfoMap	972
C.5.2.1.3	Enumeration: TargetAddressType	973
C.5.2.2	Public Member Functions	973
C.5.2.2.1	Special Member Functions	973
C.5.2.2.1.1	Destructor	973
C.5.2.2.2	Member Functions	974
C.5.2.2.2.1	AddMetaInfo	974
C.5.2.2.2.2	GetPayload	974
C.5.2.2.2.3	GetPayload	975
C.5.2.2.2.4	GetSa	975
C.5.2.2.2.5	GetTa	976
C.5.2.2.2.6	GetTaType	976
C.5.2.3	Protected Member Functions	977
C.5.2.3.1	Special Member Functions	977
C.5.2.3.1.1	Default Constructor	977
C.6	Sequence Diagrams of UDS Transport Layer Interaction	978
C.6.1	Lifecycle	978
C.6.2	UDS Request Processing	979
C.6.3	UDS Response Transmission	980
C.6.4	Channel Reestablishment	981
D	Not implemented requirements	982
D.1	Not applicable requirements	982
E	History of Constraints and Specification Items	983
E.1	Constraint and Specification Item History of this document according to AUTOSAR Release 17-10	983
E.1.1	Added Specification Items in 17-10	983
E.1.2	Changed Specification Items in 17-10	985
E.1.3	Deleted Specification Items in 17-10	987
E.1.4	Added Constraints in 17-10	988
E.1.5	Changed Constraints in 17-10	988
E.1.6	Deleted Constraints in 17-10	988
E.2	Constraint and Specification Item History of this document according to AUTOSAR Release 18-03	989
E.2.1	Added Specification Items in 18-03	989
E.2.2	Changed Specification Items in 18-03	990
E.2.3	Deleted Specification Items in 18-03	996
E.2.4	Added Constraints in 18-03	996
E.2.5	Changed Constraints in 18-03	996
E.2.6	Deleted Constraints in 18-03	997
E.3	Constraint and Specification Item History of this document according to AUTOSAR Release 18-10	997
E.3.1	Added Specification Items in 18-10	997

E.3.2	Changed Specification Items in 18-10	999
E.3.3	Deleted Specification Items in 18-10	1005
E.3.4	Added Constraints in 18-10	1006
E.3.5	Changed Constraints in 18-10	1006
E.3.6	Deleted Constraints in 18-10	1007
E.4	Constraint and Specification Item History of this document according to AUTOSAR Release 19-03	1007
E.4.1	Added Specification Items in 19-03	1007
E.4.2	Changed Specification Items in 19-03	1012
E.4.3	Deleted Specification Items in 19-03	1013
E.4.4	Added Constraints in 19-03	1013
E.4.5	Changed Constraints in 19-03	1013
E.4.6	Deleted Constraints in 19-03	1014
E.5	Constraint and Specification Item History of this document according to AUTOSAR Release R19-11	1014
E.5.1	Added Specification Items in 19-11	1014
E.5.2	Changed Specification Items in 19-11	1017
E.5.3	Deleted Specification Items in 19-11	1022
E.5.4	Added Constraints in 19-11	1025
E.5.5	Changed Constraints in 19-11	1025
E.5.6	Deleted Constraints in 19-11	1025
E.6	Constraint and Specification Item History of this document according to AUTOSAR Release R20-11	1025
E.6.1	Added Specification Items in R20-11	1025
E.6.2	Changed Specification Items in R20-11	1031
E.6.3	Deleted Specification Items in R20-11	1035
E.6.4	Added Constraints in R20-11	1035
E.6.5	Changed Constraints in R20-11	1035
E.6.6	Deleted Constraints in R20-11	1035
E.7	Constraint and Specification Item History of this document according to AUTOSAR Release R21-11	1035
E.7.1	Added Specification Items in R21-11	1035
E.7.2	Changed Specification Items in R21-11	1040
E.7.3	Deleted Specification Items in R21-11	1050
E.7.4	Added Constraints in R21-11	1051
E.7.5	Changed Constraints in R21-11	1052
E.7.6	Deleted Constraints in R21-11	1052
E.8	Constraint and Specification Item History of this document according to AUTOSAR Release R22-11	1052
E.8.1	Added Specification Items in R22-11	1052
E.8.2	Changed Specification Items in R22-11	1061
E.8.3	Deleted Specification Items in R22-11	1074
E.8.4	Added Constraints in R22-11	1075
E.8.5	Changed Constraints in R22-11	1075

E.8.6 Deleted Constraints in R22-11	1075
E.9 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11	1075
E.9.1 Added Specification Items in R23-11	1075
E.9.2 Changed Specification Items in R23-11	1086
E.9.3 Deleted Specification Items in R23-11	1092
E.9.4 Added Constraints in R23-11	1092
E.9.5 Changed Constraints in R23-11	1093
E.9.6 Deleted Constraints in R23-11	1093
E.10 Constraint and Specification Item History of this document according to AUTOSAR Release R24-11	1093
E.10.1 Added Specification Items in R24-11	1093
E.10.2 Changed Specification Items in R24-11	1098
E.10.3 Deleted Specification Items in R24-11	1143
E.10.4 Added Constraints in R24-11	1146
E.10.5 Changed Constraints in R24-11	1146
E.10.6 Deleted Constraints in R24-11	1147
E.11 Constraint and Specification Item History of this document according to AUTOSAR Release R25-11	1147
E.11.1 Added Specification Items in R25-11	1147
E.11.2 Changed Specification Items in R25-11	1159
E.11.3 Deleted Specification Items in R25-11	1170
E.11.4 Added Constraints in R25-11	1171
E.11.5 Changed Constraints in R25-11	1171
E.11.6 Deleted Constraints in R25-11	1171

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. The DoIP protocol based on ISO 13400-2:2019[2], including ISO 13400-2:2019/Amd 1:2023[3] or a custom implementation of a transport protocol can be used.

For SOVD, the [HTTP/HTTPS](#) 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) [4] is the configuration input to the DM.

1.3 Software Cluster

The AUTOSAR adaptive platform (AP) 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](#) has its own [DiagnosticAddress](#).

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 [5]. 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.

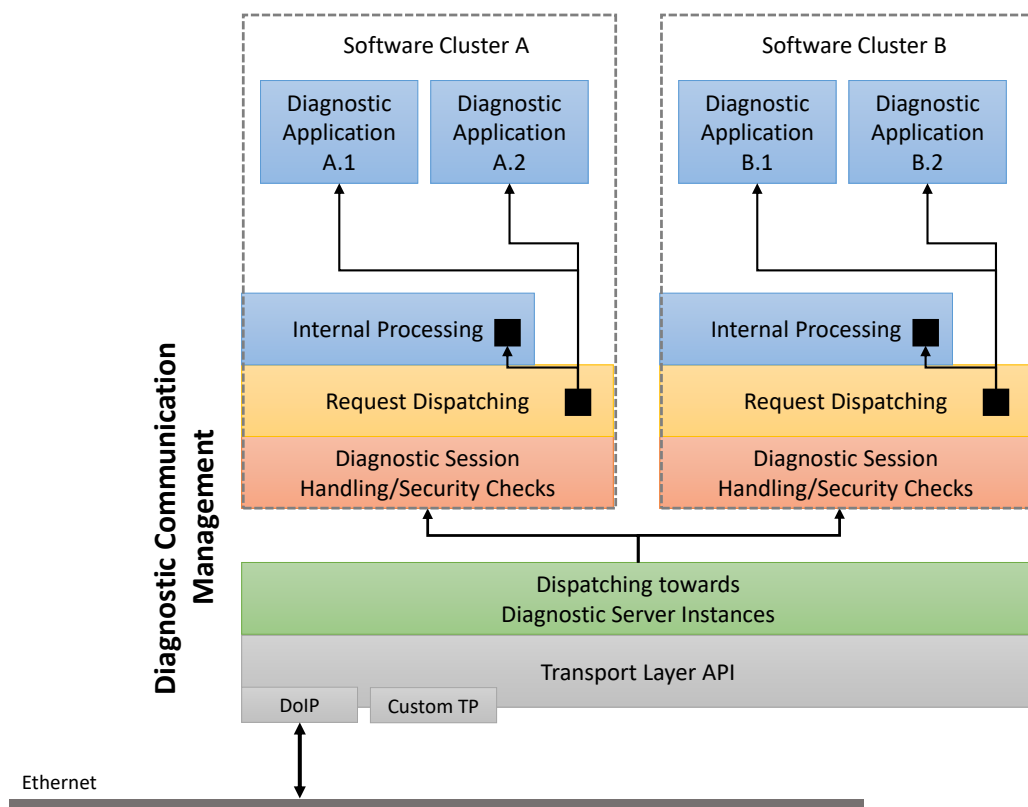


Figure 1.1: Architecture Diagnostic Communication Management

1.3.2 Diagnostic Managers external dependencies

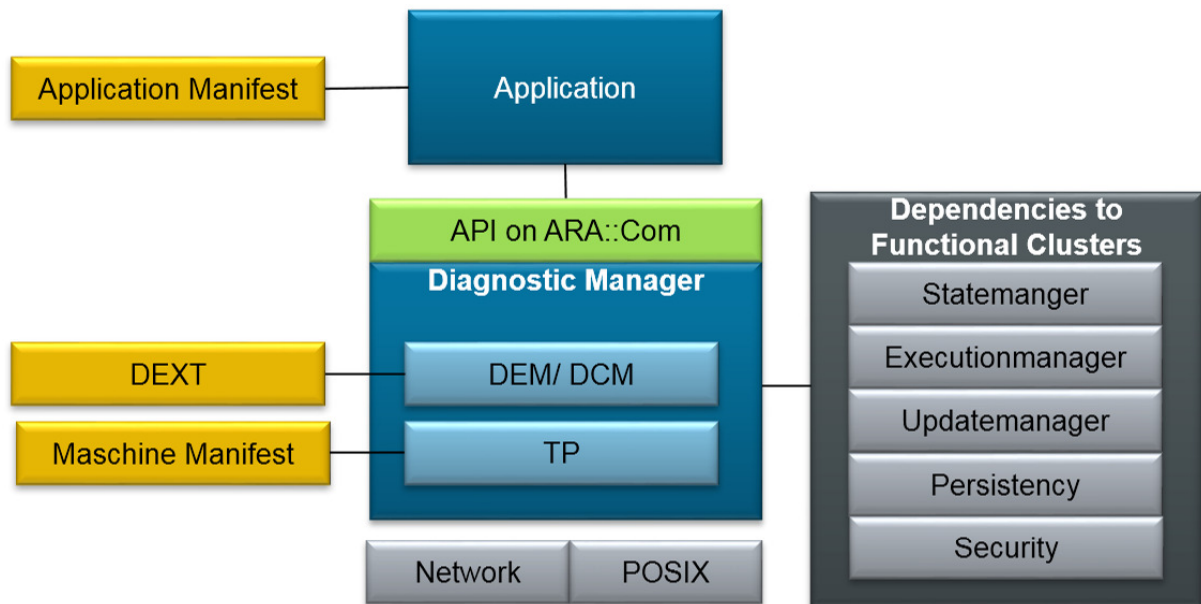


Figure 1.2: Diagnostic Managers external dependencies

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 [6].

Abbreviation / Acronym:	Description:
DCM	Diagnostic Communication Manager (Module of the AUTOSAR Classic Platform)
DDID	Dynamically Defined Data Identifier
DID	Diagnostic Identifier
DNS	Domain Name System
EDR	Extended Data Record
EID	Entity Identifier
FDC	Fault Detection Counter
GID	Group Identifier
HTTP	Hypertext-Transport-Protocol (see RFC 7540 and RFC 7541 for HTTP/2 (backward compatible to HTTP/1.1))
HTTPS	Hypertext Transfer Protocol Secure (secure variant of HTTP)
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.
Oc	Occurrence counter
OC	Operation Cycle
OCC	Operation Cycle Counter
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].
POWN	Proof of Ownership (see Authentication Service, ISO 14229-1[1])
RID	Routine Identifier
RoE	Response on Event
SA	The SourceAddress of a UDS request
SID	Service Identifier, identifying a diagnostic service according to UDS
TA	The TargetAddress of a UDS request
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VIN	Vehicle Identification Number

Table 2.1: Abbreviations & Acronyms

The following technical terms used in this document are defined in the corresponding document mentioned in the table below.

Terms:	Description:
Aging	Unlearning/deleting of a no longer failed event / DTC after a defined number of operation cycles from event memory.
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.

Terms:	Description:
Diagnostic Client	A Diagnostic Client is a diagnostic service requester, i.e. sends a UDS request to the Diagnostic Server , which is processed there and answered as a UDS response. 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	A Diagnostic Conversation represents a conversation between a Diagnostic Client (Tester) and a Diagnostic Server , using the UDS protocol, as defined in ISO 14229-1[1].
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	A 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 Diagnostic 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 , displacement is used, this means: Replacing the most insignificant stored event memory entry by a reported event which needs to be stored and is more significant.
DTC group	A 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.
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 are called Enable Conditions . The configuration is done per event .
Event	An event (also <i>diagnostic event</i>) 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.

Terms:	Description:
Event status	The Event status-byte is a bit-packed status information based on Event level. It contains the following bits: Nr. Definition: 0 testFailed 1 testFailedThisOperationCycle 6 testNotCompletedThisOperationCycle See also UDS DTC status bit.
Execution Management	Functional cluster Execution Management
Extended Data Records	Extended Data Records contain statistical data for a DTC. Extended Data Records are assigned to DTCs and maintained and stored by the DM.
Fail-safe reaction	Fail-safe reaction , sometimes also called "limp home mode": Reaction to avoid or minimize harm or damage in case of a failure.
Fault Detection Counter	The FDC according to ISO 14229-1[1] is a counter, used to determine the state of an DTC. The FDC has always the value range: FDC_{min} from -128 = "FinallyHealed" FDC_{max} to +127 = "FinallyDefective"
GroupOfAllDTCs	The GroupOfAllDTCs identifies a special DTC group that contains all DTCs. This DTC group is identified by the DTC value 0xFFFFFFFF in ISO 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 application, 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 .
MetaInfo	Meta-Information in the form of a key-value map, which is given from DM to external service processors.
Monitor	A monitor (also <i>diagnostic monitor</i>) 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 .
Negative Response Code	A Negative Response Code is used by UDS in the diagnostic response to indicate to the tester, that a certain failure has occurred and the diagnostic request was not processed. An exception is the NRC 0x78 0x78 - RequestCorrectlyReceivedResponsePending. The request was formally received correct, however, the processing requires further time. See ISO 14229-1[1] for further details.
Non-volatile Memory	In the context of the DM, Non-volatile Memory refers to the persistent information over the shutdown of the DM process. This does not depend on HW details.
Occurrence counter	The Occurrence counter is a counter, used in the area of Onboard Diagnostics .

Terms:	Description:
Onboard 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 series [7] (SAE J1979-2 OBDOnUDS, SAE J1979-3 ZEVonUDS), ISO 15031 series ([8][9]), ISO 27145 series ([10][11][12]) (WWH-OBD) and others.
Operation Cycle	A new operation cycle is the start of a new monitoring cycle. This is reflected in a reset of the <code>testFailedThisOperationCycle</code> and <code>testNotCompletedThisOperationCycle</code> bits in the DTC status and optional notification to the <code>monitor</code> to restart the monitoring. The <code>OCC</code> represents the number of <code>Operation Cycles</code> .
OpenAPI specification	Specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services. These files are used by SOVD as offline capability description, see also ISO 17978-3[13] and related standards (e.g. OpenAPI, ASAM SOVD[5][14]).
Primary event memory	The primary event memory is used to store events and event related data. It is typically used by <code>OEMs</code> for after sales purposes, containing information to repair the vehicle.
PowerMode	Vehicle basic status information retrieval of DoIP (ISO 13400-2[2])
Snapshot Record	Snapshots (sometimes referred to as <code>freeze frames</code>) 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 <code>SoftwareCluster</code> 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 <code>SoftwareCluster</code> can be addressed individually by its own set of diagnostic addresses.
Source Address	A <code>Source Address</code> is used to encode client and server identifiers. Request In a UDS request, the <code>source address</code> encodes the <code>Diagnostic Client</code>. Response In a UDS response, the <code>source address</code> encodes the <code>Diagnostic Server</code>.
SOVD client	A <code>HTTP</code> based diagnostic tester, implementing ISO 17978-3[13] (ASAM SOVD 1.1.0[14]), is a <code>SOVD client</code> .
SOVD entity	Entity in context of SOVD[5]
SOVD lock	SOVD locking mechanism as defined by [5]
Target Address	A <code>Target Address</code> is used to encode client and server identifiers. Request In a UDS request, the <code>target address</code> encodes the the <code>Diagnostic Server</code>. Response In a UDS response, the <code>target address</code> encodes the the <code>Diagnostic Client</code>.
Transport Protocol Handler	A sub-component 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]

Terms:	Description:
UDS DTC status bit	The UDS DTC status bits are 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. See also 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.
Vehicle Identification Number	The VIN is defined in ISO-3779. The UDS specification ISO 14229-1[1] Annex C.1 defines the DID F190 as the one to reference the VIN number. The record data is vehicle manufacturer specific, however, an array of 17 uint8, typically in the ASCII range, is quite common.

Table 2.2: Technical Terms

The following technical terms used in this document are defined in the corresponding document mentioned in the table below.

Terms:	Description:
Diagnostic Identifier	see ISO 14229-1[1], see RDBI Service (\$22), WDBI (\$2E), IOC (\$2F), DDDI and related Services. This 16 bit value uniquely defines one or more data elements (parameters) that can be used in diagnostics to read, write or control data.
Dynamically Defined Data Identifier	see ISO 14229-1[1], Chapter 11.6, Dynamically Defined Data Identifier Service (\$2C)
Entity Identifier	see ISO 13400-2[2]
Group Identifier	see ISO 13400-2[2]
Response on Event	see ISO 14229-1[1], Chapter 10.9, Response on Event Service (\$86)
Routine Identifier	see ISO 14229-1[1], Chapter 14, Routine Control Service (\$31)
Test	Test.

Table 2.3: Reference to Technical Terms

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 (Edition 2, Release 2019-12)
<https://www.iso.org/standard/74785.html>
- [3] ISO 13400-2:2019/Amd 1:2023 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Edition 2, Amendment 1, Release 2023-07)
<https://www.iso.org/standard/84208.html>
- [4] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate
- [5] ASAM Service Oriented Vehicle Diagnostics V1.0.0
<http://www.asam.net>
ASAM_SOVD_BS_V1-0-0_PR.pdf
- [6] Glossary
AUTOSAR_FO_TR_Glossary
- [7] SAE J1979 – E/E Diagnostic Test Modes
- [8] ISO 15031-5 – Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 5: Emission-related diagnostic services.
<https://www.iso.org/standard/66368.html>
- [9] ISO 15031-6 – Road vehicles – Communication between vehicle and external equipment for emission-related diagnostic – Part 6: Diagnostic trouble code definitions
<https://www.iso.org>
- [10] ISO 27145-1:2012 – Road vehicles – Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD) communication requirements – Part 1: General information and use case definition
<https://www.iso.org/standard/46273.html>
- [11] ISO 27145-2:2012 – Road vehicles – Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD) communication requirements – Part 2: Common data dictionary
<https://www.iso.org/standard/46276.html>
- [12] ISO 27145-3:2012 – Road vehicles – Implementation of World-Wide Harmonized

On-Board Diagnostics (WWH-OBD) communication requirements – Part 3: Common message dictionary
<https://www.iso.org/standard/46277.html>

- [13] ISO 17978-3:2025 – Road vehicles — Service-oriented vehicle diagnostics (SOVD) – Part 3: Application programming interface (API)
<http://www.iso.org>
- [14] ASAM Service Oriented Vehicle Diagnostics V1.1.0
<http://www.asam.net>
ASAM_SOVD_BS_V1-1-0_PR.pdf
- [15] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core
- [16] ISO 27145-2:2012/Amd 1:2023 – Road vehicles – Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD) communication requirements – Part 2: Common data dictionary
<https://www.iso.org/standard/86149.html>
- [17] ISO 13400-2:2025 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services (Edition 3, Release 2025-05)
<https://www.iso.org/standard/87961.html>
- [18] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture
- [19] Requirements on Diagnostics
AUTOSAR_FO_RS_Diagnostics
- [20] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General
- [21] ISO 15765-2 – Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2: Network layer services
- [22] ISO 13400 – Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
<https://www.iso.org>
- [23] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification
- [24] IP Version 6 Addressing Architecture
<https://www.rfc-editor.org/rfc/rfc4291.txt>
<https://www.rfc-editor.org/rfc/rfc4291.txt>
- [25] ISO 14229-2(2021) – Unified diagnostic services (UDS) - Part 2: Session layer services (Release 2021-10)
<https://www.iso.org>

- [26] Requirements on Intrusion Detection System
AUTOSAR_FO_RS_IntrusionDetectionSystem
- [27] Specification of Language Binding for modeled AP data types
AUTOSAR_AP_SWS_LanguageBindingForModeledAPdatatypes
- [28] Specification of Platform Types for Adaptive Platform
AUTOSAR_AP_SWS_PlatformTypes
- [29] ISO/IEC 14882:2014, Information technology – Programming languages – C++
<https://www.iso.org>
- [30] ISO 14229-1 – Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2006-12)
<https://www.iso.org>
- [31] SAE J1939-73 Application Layer – Diagnostics
- [32] 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 [15] which is also applicable for this functional cluster. The chapter [15] 7.1 “*General requirements for all Functional Clusters*” 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 limitations 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.

4.1.1 UDS support

The following international standards are not supported by the DM:

- Legislated OBD services (ISO 15031-5:2015[8]) are not supported by the DM.
- Legislated WWH-OBD services (ISO 27145-2:2012[11], ISO 27145-2:2012 Amd 1:2023[16], ISO 27145-3:2012[12]) are not supported by the DM.

The Sub-functions of [UDS services](#) are implemented according to ISO 14229-1:2020[1] (Edition 3) unless explicitly stated.

The following [UDS services](#) are not implemented by the DM:

- 0x23 ReadMemoryByAddress
- 0x24 ReadScalingDataByIdentifier
- 0x2F InputOutputControlByIdentifier
- 0x3D WriteMemoryByAddress
- 0x83 AccessTimingParameter
- 0x84 SecuredDataTransmission
- 0x87 LinkControl

The following [UDS services](#) have limitations as listed below:

- 0x19 ReadDTCInformation:
The following sub-functions are not supported:

Sub-Function	Hint
0x05 (reportDTCStoredDataByRecordNumber)	1
0x08 (reportDTCBySeverityMaskRecord)	2
0x09 (reportSeverityInformationOfDTC)	2
0x0B (reportFirstTestFailedDTC)	2





0x0C (reportFirstConfirmedDTC)	2
0x0D (reportMostRecentTestFailedDTC)	2
0x0E (reportMostRecentConfirmedDTC)	2
0x0F (reportMirrorMemoryDTCByStatusMask)	3
0x10 (reportMirrorMemoryDTCExtDataRecordByDTCNumber)	3
0x11 (reportNumberOfMirrorMemoryDTCByStatusMask)	3
0x12 (reportNumberOfEmissionsOBDDTCByStatusMask)	1
0x13 (reportEmissionsOBDDTCByStatusMask)	1
0x15 (reportDTCWithPermanentStatus)	1
0x16 (reportDTCExtDataRecordByRecordNumber)	2
0x42 (reportWWHOBDDTCByMaskRecord)	1
0x55 (reportWWHOBDDTCWithPermanentStatus)	1

Table 4.1: Limitations of sub-functions of 0x19 ReadDTCInformation

The table lists the limitations of the sub-functions of 0x19 ReadDTCInformation

Hints:

- 1.) OBD/WWH-OBD is not supported by the DM.
- 2.) The sub-function is generally not supported by the DM.
- 3.) The UDS feature *mirror event memory* is not supported by the DM.

- 0x27 SecurityAccess:

- Handling of security access delay timer:
The behavior of the *Delay_Timer*, as described in ISO 14229-1:2020[1] chapter 10.4.1 and table I.1 "State transitions - parameters", that, after a power-up, a single false SecurityAccess attempt in the previous power-cycle already starts the *Delay_Timer*, is not supported by the DM. Instead, [SWS_DM_00479] is followed.
- 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. Selecting the *persistent storage* is mandatory in Adaptive Autosar.

- 0x29 Authentication:

- The DM only implements *Authentication with PKI Certificate Exchange* (APCE), as described in ISO 14229-1:2020[1] chapter 10.6.2.
- *Authentication with Challenge-Response* (ACR), as described in ISO 14229-1:2020[1] chapter 10.6.3, is currently out of scope.

- 0x2A ReadDataByPeriodicIdentifier: Only *scheduler type 1* is supported.

- 0x2C DynamicallyDefineDataIdentifier:
 - Subfunction 0x02 *defineByMemoryAddress* is not supported.
- 0x85 ControlDTCSetting:
 - Only subfunctions 0x01 (ON) and 0x02 (OFF) are supported.
- 0x86 ResponseOnEvent:
 - Queuing of *events* is not supported.
 - Regarding the request message parameter *eventWindowTime* (refer to Annex B.2 in ISO 14229-1[1]), the only values supported are *infiniteTimeToResponse* and *powerWindowTime*.

Storage:

- Event Memory: Variant handling at runtime for events/DTCs is not supported.
- Event Memory: Details for combined events are not specified.

Special Service Checks:

- Manufacturer- / 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.

NRC codes (Annex A.1 "Negative response codes") of ISO 14229-1[1]:

- OEM-specific error codes are by intention not part of the AUTOSAR standardization but can be added platform vendor specific as an additional ErrorDomain.

4.1.2 DoIP support

The following limitations apply for the ISO 13400-2 (DoIP) series:

- *DoIP gateway* is not supported by the DM, this means, no further routing to underlying sub-networks is foreseen and related DoIP-requirements, do not apply.
- Value 0x00 is not supported as default value for *EID*, *GID* and *VIN* as specified in ISO 13400-2[2]
- Manufacturer-specific-PayloadType and DoIP-PayloadType 0x9001 as defined in ISO 13400-2:2025[17] (Edition 3), are not supported.

4.1.3 SOVD support

For ASAM SOVD 1.0 (2022-07) [5] 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 capabilities 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.
- The SOVD attribute `symptom` of a SOVD fault is not supported.
- The serialization of SOVD data, with `DiagnosticSovd-Props.singleElementSerialization` set to `plain` is not supported.

For ASAM SOVD 1.1 (2026-01) [14] / ISO 17978-3:2026 [13] the following limitations apply:

- The SOVD API based on version 1.1 / ISO 17978-3:2026 is not yet (fully) supported.
- Bugfixes between SOVD 1.0 and 1.1 have been introduced to this document, based on the ISO 17978-3 (DIS/FDIS) document.
- Changes in SOVD 1.1, which have been done to comply to ISO guidelines, will be introduced after official release of the ISO 17978-3, e.g. "Stop the execution of an operation" is "Terminate the execution of an operation".

4.1.4 General

Software Cluster / Diagnostic Service Instances:

- *Software Cluster/Diagnostic Server instances* are supported by DM interfaces but are not specified in detail.

4.1.5 AP stabilization

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 [18].

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.

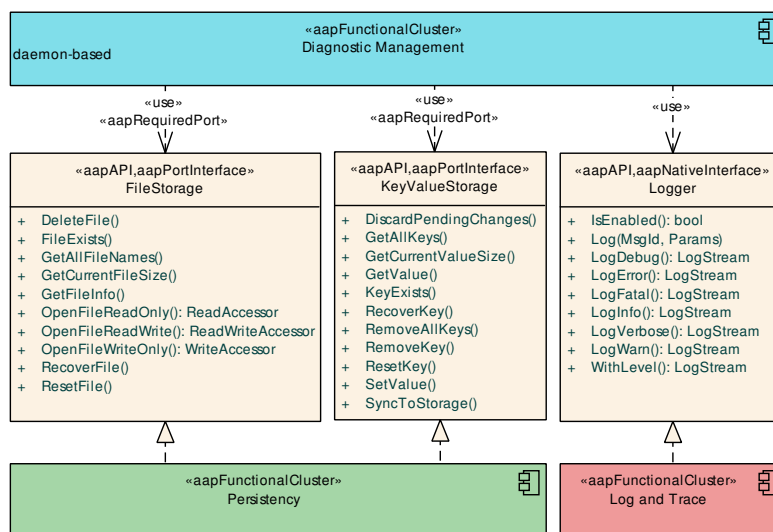


Figure 5.1: Interfaces required by Diagnostic Management

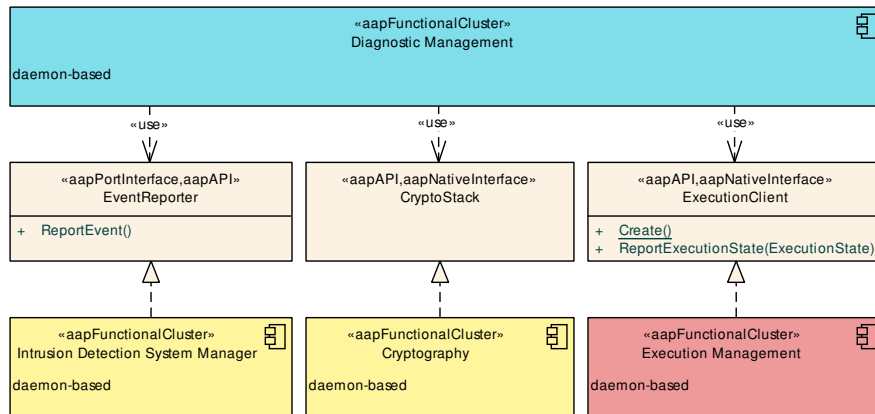


Figure 5.2: Interface required by Diagnostic Management from Adaptive Intrusion Detection System Manager

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 <i>Diagnostic Management</i> daemon process(es).
Intrusion Detection System Manager	EventReporter	<i>Diagnostic Management</i> uses this interface to report standardized security events.
Log and Trace	Logger	<i>Diagnostic Management</i> 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	<i>Diagnostic Management</i> should use this interface to enable supervision of its daemon process(es) by <i>Platform Health Management</i> .

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in Diagnostics [19] and the AUTOSAR RS General [20], 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]	Compatibility with the ISO 14882 C++ standard	[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_02154] [SWS_DM_02156] [SWS_DM_02157] [SWS_DM_02159] [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]





Requirement	Description	Satisfied by
		△ [SWS_DM_01709] [SWS_DM_01806] [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] [SWS_DM_02152]
[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] [SWS_DM_02121]
[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 <code>ara::core</code> 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] [SWS_DM_02154] [SWS_DM_02156]
[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] [SWS_DM_02154] [SWS_DM_02156]
[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] [SWS_DM_02090] [SWS_DM_02091] [SWS_DM_02092] [SWS_DM_02093] [SWS_DM_02151]





Requirement	Description	Satisfied by
[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] [SWS_DM_02121] [SWS_DM_02158]
[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] [SWS_DM_02154] [SWS_DM_02156]
[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_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]





Requirement	Description	Satisfied by
		△ [SWS_DM_02076] [SWS_DM_02078] [SWS_DM_02118] [SWS_DM_02122] [SWS_DM_02123] [SWS_DM_02124] [SWS_DM_02125] [SWS_DM_02126] [SWS_DM_02127] [SWS_DM_02128] [SWS_DM_02129] [SWS_DM_02152]
[RS_AP_00142]	Handling of unsuccessful operations	[SWS_DM_01988]
[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] [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]





Requirement	Description	Satisfied by
		△ [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] [SWS_DM_02147] [SWS_DM_02148] [SWS_DM_02149] [SWS_DM_02150]
[RS_AP_00149]	Error handling for non-initialized Functional Cluster	[SWS_DM_02061]
[RS_AP_00170]	InstanceSpecifierMappingIntegrity Violation	[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_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_00831] [SWS_DM_01010] [SWS_DM_01124] [SWS_DM_01193] [SWS_DM_01325] [SWS_DM_01364] [SWS_DM_01480] [SWS_DM_01489] [SWS_DM_01695] [SWS_DM_01705] [SWS_DM_01727] [SWS_DM_01823] [SWS_DM_01847] [SWS_DM_01864] [SWS_DM_01910] [SWS_DM_01962] [SWS_DM_01972]
[RS_AP_00171]	PortInterfaceMappingViolation	[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_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_00831] [SWS_DM_01010] [SWS_DM_01124] [SWS_DM_01193] [SWS_DM_01325] [SWS_DM_01364] [SWS_DM_01480] [SWS_DM_01489] [SWS_DM_01695] [SWS_DM_01705] [SWS_DM_01727] [SWS_DM_01823] [SWS_DM_01847] [SWS_DM_01864] [SWS_DM_01910] [SWS_DM_01962] [SWS_DM_01972]





Requirement	Description	Satisfied by
[RS_AP_00172]	ProcessMappingViolation	[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_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_00831] [SWS_DM_01010] [SWS_DM_01124] [SWS_DM_01193] [SWS_DM_01325] [SWS_DM_01364] [SWS_DM_01480] [SWS_DM_01489] [SWS_DM_01695] [SWS_DM_01705] [SWS_DM_01727] [SWS_DM_01823] [SWS_DM_01847] [SWS_DM_01864] [SWS_DM_01910] [SWS_DM_01962] [SWS_DM_01972]
[RS_AP_00173]	InstanceSpecifierAlreadyInUse Violation	[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_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_00831] [SWS_DM_01010] [SWS_DM_01124] [SWS_DM_01193] [SWS_DM_01325] [SWS_DM_01364] [SWS_DM_01480] [SWS_DM_01489] [SWS_DM_01695] [SWS_DM_01705] [SWS_DM_01727] [SWS_DM_01823] [SWS_DM_01847] [SWS_DM_01864] [SWS_DM_01910] [SWS_DM_01962] [SWS_DM_01972]
[RS_Diag_00025]	Valid DoIP messages shall be recognized	[SWS_DM_00475] [SWS_DM_01986] [SWS_DM_02111]
[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_02005] [SWS_DM_02006] [SWS_DM_02007] [SWS_DM_02008] [SWS_DM_02009] [SWS_DM_02010] [SWS_DM_02107]
[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]





Requirement	Description	Satisfied by
[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]
[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_00363] [SWS_DM_00421] [SWS_DM_00478] [SWS_DM_00479] [SWS_DM_00480] [SWS_DM_00507] [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_00863] [SWS_DM_00864] [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_00134] [SWS_DM_00137] [SWS_DM_00141] [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_00867] [SWS_DM_00868] [SWS_DM_00869] [SWS_DM_00871] [SWS_DM_01096] [SWS_DM_01097]





Requirement	Description	Satisfied by
[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]
[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] [SWS_DM_02080] [SWS_DM_02081] [SWS_DM_02082] [SWS_DM_02083] [SWS_DM_02084] [SWS_DM_02085] [SWS_DM_02086] [SWS_DM_02087] [SWS_DM_02088]
[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] [SWS_DM_01971] [SWS_DM_02061]
[RS_Diag_04071]	Process events according to their defined importance like priority and/or severity	[SWS_DM_00916]
[RS_Diag_04091]	Application access to snapshot record data	[SWS_DM_00660] [SWS_DM_00661] [SWS_DM_00662] [SWS_DM_00668] [SWS_DM_00894] [SWS_DM_02112] [SWS_DM_02113] [SWS_DM_02115] [SWS_DM_02116] [SWS_DM_02117] [SWS_DM_02123]





Requirement	Description	Satisfied by
[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_02145]
[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] [SWS_DM_02124]
[RS_Diag_04115]	The optional parameter DTCSettingControlOptionRecord 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]
[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]
[RS_Diag_04120]	Support a predefined AddressAndLengthFormatIdentifier	[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 DTCSnapshotRecords and DTCExtendedDataRecords	[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]





Requirement	Description	Satisfied by
[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] [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]





Requirement	Description	Satisfied by
[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_01573] [SWS_DM_02382] [SWS_DM_02387] [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] [SWS_DM_02122]
[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]
[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_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] [SWS_DM_02125]





Requirement	Description	Satisfied by
[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] [SWS_DM_01978]
[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_00843] [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] [SWS_DM_02106] [SWS_DM_02108] [SWS_DM_02109] [SWS_DM_02110] [SWS_DM_02182] [SWS_DM_02183] [SWS_DM_02184] [SWS_DM_02185]
[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]
[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_01573] [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]





Requirement	Description	Satisfied by
[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] [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_02126] [SWS_DM_02127] [SWS_DM_02128] [SWS_DM_02146] [SWS_DM_02151] [SWS_DM_02152] [SWS_DM_02153] [SWS_DM_02155] [SWS_DM_80003] [SWS_DM_80004] [SWS_DM_80005] [SWS_DM_80013] [SWS_DM_80023]





Requirement	Description	Satisfied by
[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] [SWS_DM_02094] [SWS_DM_02095] [SWS_DM_02096] [SWS_DM_02097] [SWS_DM_02098] [SWS_DM_02099] [SWS_DM_02102] [SWS_DM_02103] [SWS_DM_02104] [SWS_DM_02105]
[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]
[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] [SWS_DM_01987] [SWS_DM_02062] [SWS_DM_02121]





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_00113] [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] [SWS_DM_02118]
[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]
[RS_Diag_04197]	Clearing the user defined fault memory	[SWS_DM_00193] [SWS_DM_00194] [SWS_DM_00195] [SWS_DM_00208]





Requirement	Description	Satisfied by
[RS_Diag_04198]	Process all UDS Services related to session and security management of ISO 14229 internally	[SWS_DM_00126] [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]
[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] [SWS_DM_01976] [SWS_DM_01977]
[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]
[RS_Diag_04204]	Provide the current status of each warning indicator.	[SWS_DM_00223] [SWS_DM_00224] [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_01266] [SWS_DM_01359] [SWS_DM_01582] [SWS_DM_01974] [SWS_DM_01975] [SWS_DM_02129]
[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] [SWS_DM_02112] [SWS_DM_02113] [SWS_DM_02114] [SWS_DM_02115] [SWS_DM_02116] [SWS_DM_02117] [SWS_DM_02123] [SWS_DM_02142]
[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] [SWS_DM_02127] [SWS_DM_02128]
[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_01571] [SWS_DM_01574] [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]





Requirement	Description	Satisfied by
[RS_Diag_04215]	Support of UDS service ReadDataByPeriodicIdentifier (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_02152] [SWS_DM_02155] [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]





Requirement	Description	Satisfied by
[RS_Diag_04228]	Common check for Message lengths	[SWS_DM_00098] [SWS_DM_00412] [SWS_DM_00507] [SWS_DM_01041] [SWS_DM_01042]
[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_02100] [SWS_DM_02101]
[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] [SWS_DM_01571] [SWS_DM_01574]
[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_01394] [SWS_DM_01403] [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] [SWS_DM_02161] [SWS_DM_02162] [SWS_DM_02163] [SWS_DM_02167] [SWS_DM_02170] [SWS_DM_02178]
[RS_Diag_04259]	Fault Memory Access	[SWS_DM_01406] [SWS_DM_01409] [SWS_DM_01411] [SWS_DM_01412] [SWS_DM_01414] [SWS_DM_01415] [SWS_DM_01416] [SWS_DM_01417] [SWS_DM_01418] [SWS_DM_01561] [SWS_DM_01567] [SWS_DM_02000] [SWS_DM_02001] [SWS_DM_02119] [SWS_DM_02120]
[RS_Diag_04260]	Read / Write Access	[SWS_DM_01394] [SWS_DM_01419] [SWS_DM_01420] [SWS_DM_01431] [SWS_DM_01432] [SWS_DM_01433] [SWS_DM_01442] [SWS_DM_01446] [SWS_DM_01448] [SWS_DM_01449] [SWS_DM_02160] [SWS_DM_02161] [SWS_DM_02162] [SWS_DM_02163] [SWS_DM_02164] [SWS_DM_02165] [SWS_DM_02166] [SWS_DM_02178] [SWS_DM_02186] [SWS_DM_02187] [SWS_DM_02188] [SWS_DM_02189] [SWS_DM_02190] [SWS_DM_02191] [SWS_DM_02192] [SWS_DM_02193] [SWS_DM_02194] [SWS_DM_02213] [SWS_DM_02214] [SWS_DM_02215] [SWS_DM_02216] [SWS_DM_02217] [SWS_DM_02218] [SWS_DM_02219] [SWS_DM_02220] [SWS_DM_02221] [SWS_DM_02222] [SWS_DM_02223] [SWS_DM_02224] [SWS_DM_02225] [SWS_DM_02226] [SWS_DM_02227] [SWS_DM_02228] [SWS_DM_02229] [SWS_DM_02230] [SWS_DM_02231] [SWS_DM_02232] [SWS_DM_02233] [SWS_DM_02234] [SWS_DM_02235] [SWS_DM_02236] [SWS_DM_02237]





Requirement	Description	Satisfied by
		 [SWS_DM_02238] [SWS_DM_02239] [SWS_DM_02240] [SWS_DM_02241] [SWS_DM_02242] [SWS_DM_02243] [SWS_DM_02244] [SWS_DM_02245] [SWS_DM_02246] [SWS_DM_02247] [SWS_DM_02248] [SWS_DM_02249] [SWS_DM_02250] [SWS_DM_02251] [SWS_DM_02252] [SWS_DM_02253] [SWS_DM_02254] [SWS_DM_02255] [SWS_DM_02256] [SWS_DM_02257] [SWS_DM_02258] [SWS_DM_02259] [SWS_DM_02260] [SWS_DM_02261] [SWS_DM_02262] [SWS_DM_02263] [SWS_DM_02264] [SWS_DM_02265] [SWS_DM_02266] [SWS_DM_02267] [SWS_DM_02268] [SWS_DM_02269] [SWS_DM_02270] [SWS_DM_02271] [SWS_DM_02272] [SWS_DM_02273] [SWS_DM_02274] [SWS_DM_02275] [SWS_DM_02276] [SWS_DM_02277] [SWS_DM_02278] [SWS_DM_02279] [SWS_DM_02280] [SWS_DM_02281] [SWS_DM_02282] [SWS_DM_02283] [SWS_DM_02284] [SWS_DM_02285] [SWS_DM_02286] [SWS_DM_02287] [SWS_DM_02288] [SWS_DM_02289] [SWS_DM_02290] [SWS_DM_02291] [SWS_DM_02292] [SWS_DM_02293] [SWS_DM_02294] [SWS_DM_02295] [SWS_DM_02296] [SWS_DM_02297] [SWS_DM_02298] [SWS_DM_02299] [SWS_DM_02300] [SWS_DM_02301] [SWS_DM_02302] [SWS_DM_02303] [SWS_DM_02304] [SWS_DM_02305] [SWS_DM_02306] [SWS_DM_02307] [SWS_DM_02308] [SWS_DM_02309] [SWS_DM_02310] [SWS_DM_02311] [SWS_DM_02312] [SWS_DM_02313] [SWS_DM_02314] [SWS_DM_02315] [SWS_DM_02316] [SWS_DM_02317] [SWS_DM_02318] [SWS_DM_02319] [SWS_DM_02320] [SWS_DM_02321] [SWS_DM_02322] [SWS_DM_02323] [SWS_DM_02324] [SWS_DM_02325] [SWS_DM_02326] [SWS_DM_02327] [SWS_DM_02328] [SWS_DM_02329] [SWS_DM_02330] [SWS_DM_02331] [SWS_DM_02332] [SWS_DM_02333] [SWS_DM_02334] [SWS_DM_02335] [SWS_DM_02336] [SWS_DM_02337] [SWS_DM_02338] [SWS_DM_02339] [SWS_DM_02340] [SWS_DM_02341] [SWS_DM_02342] [SWS_DM_02343] [SWS_DM_02344] [SWS_DM_02345] [SWS_DM_02346] [SWS_DM_02347] [SWS_DM_02348] [SWS_DM_02349] [SWS_DM_02350] [SWS_DM_02351] [SWS_DM_02352] [SWS_DM_02353] [SWS_DM_02354] [SWS_DM_02355] [SWS_DM_02356] [SWS_DM_02357] [SWS_DM_02358] [SWS_DM_02359] [SWS_DM_02360] [SWS_DM_02361] 





Requirement	Description	Satisfied by
		△ [SWS_DM_02362] [SWS_DM_02363] [SWS_DM_02364] [SWS_DM_02365] [SWS_DM_02366] [SWS_DM_02367] [SWS_DM_02368] [SWS_DM_02369] [SWS_DM_02370] [SWS_DM_02371] [SWS_DM_02372] [SWS_DM_02373] [SWS_DM_02374] [SWS_DM_02375] [SWS_DM_02376] [SWS_DM_02377] [SWS_DM_02378] [SWS_DM_02379] [SWS_DM_02380] [SWS_DM_02381] [SWS_DM_02385] [SWS_DM_02386]
[RS_Diag_04261]	Configuration Access	[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]
[RS_Diag_04262]	Control of Operations	[SWS_DM_01390] [SWS_DM_01391] [SWS_DM_01392] [SWS_DM_01393] [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] [SWS_DM_02167] [SWS_DM_02170] [SWS_DM_02171] [SWS_DM_02172] [SWS_DM_02173] [SWS_DM_02174] [SWS_DM_02175] [SWS_DM_02176] [SWS_DM_02177] [SWS_DM_02179] [SWS_DM_02180] [SWS_DM_02181] [SWS_DM_02195] [SWS_DM_02196] [SWS_DM_02197] [SWS_DM_02198] [SWS_DM_02199] [SWS_DM_02200] [SWS_DM_02201] [SWS_DM_02202] [SWS_DM_02203] [SWS_DM_02204] [SWS_DM_02205] [SWS_DM_02206] [SWS_DM_02207] [SWS_DM_02208] [SWS_DM_02209] [SWS_DM_02210] [SWS_DM_02211] [SWS_DM_02212] [SWS_DM_02383] [SWS_DM_02384]
[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]





Requirement	Description	Satisfied by
[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]
[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]





Requirement	Description	Satisfied by
[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_Diag_04275]	Support of UDS service 0x22 Read DataByIdentifier	[SWS_DM_00170] [SWS_DM_00409] [SWS_DM_00412] [SWS_DM_00413] [SWS_DM_00414]
[RS_Diag_04276]	General validation of UDS requests according to ISO 14229-1	[SWS_DM_00096] [SWS_DM_00097] [SWS_DM_00098] [SWS_DM_00099] [SWS_DM_00100] [SWS_DM_00101] [SWS_DM_00102] [SWS_DM_00103] [SWS_DM_00450]
[RS_Diag_04277]	UDS Response types	[SWS_DM_00368] [SWS_DM_00369] [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_01257] [SWS_DM_01258] [SWS_DM_02059] [SWS_DM_02060]
[RS_Diag_04278]	UDS service CommunicationControl (SID 0x28)	[SWS_DM_00140] [SWS_DM_00199] [SWS_DM_00252] [SWS_DM_00804] [SWS_DM_00805] [SWS_DM_00806] [SWS_DM_00807] [SWS_DM_00808] [SWS_DM_00809] [SWS_DM_00810] [SWS_DM_00866] [SWS_DM_01562] [SWS_DM_01563] [SWS_DM_01564]
[RS_Diag_04279]	UDS service EcuReset (SID 0x11)	[SWS_DM_00234] [SWS_DM_00235] [SWS_DM_00269] [SWS_DM_00360] [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_01309] [SWS_DM_01346] [SWS_DM_01347]
[RS_Diag_04280]	UDS service WriteDataByIdentifier (SID 0x2E)	[SWS_DM_00186] [SWS_DM_00415] [SWS_DM_00416] [SWS_DM_00417]
[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]



△

Requirement	Description	Satisfied by
		△ [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] [SWS_DM_02130] [SWS_DM_02131] [SWS_DM_02132] [SWS_DM_02133] [SWS_DM_02134] [SWS_DM_02135] [SWS_DM_02136] [SWS_DM_02137] [SWS_DM_02138] [SWS_DM_02139] [SWS_DM_02140] [SWS_DM_02141]

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 [\[5\]](#). 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[\[5\]](#). 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 [Chapter 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 [Chapter 7.1](#) “[UDS Transport Layer](#)”) that handles the connection to the [Diagnostic Client](#). Accordingly, the dispatching between SOVD Transport Layer and different SOVD sub-components need to be handled.

A broad subcomponent view on DM is given as follows:

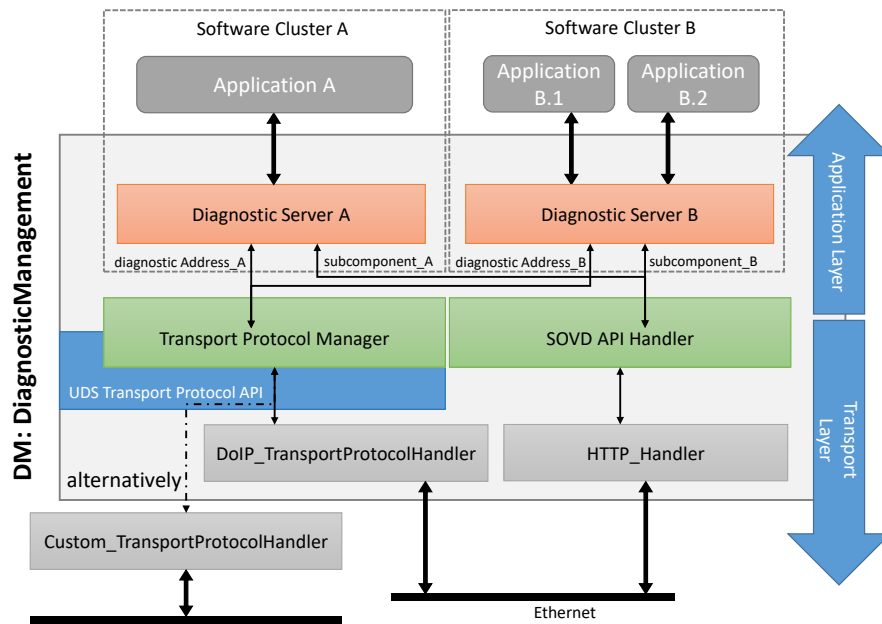


Figure 7.1: Component view on Diagnostic Management

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 15765-2[21]) 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 `suppressPosRespMsgIndicationBit = 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 [7.3.2.8.21](#).

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) resources*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](#)

[If `DoIpInstantiation.gid` is configured for a machine, the DM shall support DoIP based on the configured `DoIpFunctionalClusterDesign.doIpProtocolVersion`. For this configuration

- a value of 2 shall use ISO 13400-2:2012[\[22\]](#).
- a value of 3 shall use ISO 13400-2:2019[\[2\]](#).
- a value of 4 shall use ISO 13400-2:2019 Amendment1[\[3\]](#), ISO 13400-2:2025[\[17\]](#) and all future releases of ISO 13400-2 using the DoIP protocol version 4.

]

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

ISO 13400-2[\[2\]](#) protocol versions are mostly backward compatible. However there are extensions such as 'Negative Acknowledge' (NACK) codes or further payload types that are only available with a certain protocol version. Therefore the DM adapts the behavior based on the configured protocol version and uses for each selected protocol version only the specified values and messages from that release of ISO 13400-2[\[2\]](#).

[SWS_DM_02111] DoIP protocol semantics based on the configured DoIP protocol version

Upstream requirements: [RS_Diag_00025](#)

[The DM shall adapt the DoIP protocol behavior based on the configured [DoIpFunctionalClusterDesign.doIpProtocolVersion](#) value and shall use only DoIP messages (values, payload types, ...) that are specified by the selected ISO 13400-2[2] protocol version.]

[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 request message
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
0x8004	Periodic diagnostic message (introduced with ISO 13400-2:2019/Amd 1:2023[3])

]

[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

Upstream requirements: [RS_Diag_00026](#)

[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

Upstream requirements: [RS_Diag_00026](#)

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

[SWS_DM_02107] Vehicle Announcement/Identification Response Message without "`VIN/GID` sync status" byte

Upstream requirements: [RS_Diag_00026](#)

[If `DoIpNetworkConfiguration.vehicleIdentificationSyncStatus` is not configured or set to false, the DM shall send the Vehicle Announcement/Identification Response Message without the optional "`VIN/GID` sync status" parameter.]

[SWS_DM_02010] Vehicle Announcement/Identification Response Message with "`VIN/GID` sync status" byte

Upstream requirements: [RS_Diag_00026](#)

[If `DoIpNetworkConfiguration.vehicleIdentificationSyncStatus` is set to true, the DM shall provide the "`VIN/GID` sync status" according to ISO 13400-2[2][3].]

[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::DoIPGroupIdentification` 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] Triggering vehicle announcement messages

Upstream requirements: [RS_Diag_04242](#)

[If `ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement` is called, the DM shall trigger vehicle announcement messages according to ISO 13400-2[2] on the interface identified via `networkInterfaceId`.]

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

Upstream requirements: [RS_Diag_04242](#)

[If the DM detects an activation line change to active via the interface `ara::diag::DoIPActivationLine`, the DM shall send the vehicle announcement messages according to ISO 13400-2[2].]

The DM needs to instantiate a singleton of `ara::diag::DoIPTriggerVehicleAnnouncement` only. The `ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement` method gets the according `networkInterfaceId` 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 `networkInterfaceId`.

About "DiagnosticDoIPTriggerVehicleAnnouncementInterface"

As mentioned in [SWS_DM_00816], the DM is responsible to send the "Vehicle Announcement Message" upon detection of an established link on the network interface. The mechanism "TriggerVehicleAnnouncement" is foreseen as DM internal functionality, which depends solely on the two scenarios, as defined in the ISO 13400-2:

- send Vehicle Announcement Message on network startup (broadcast to other IP-based ECUs, Gateways, etc).
- send Vehicle Announcement Message on connection of an external tester (detection of connected IP-based ECUs).

This specification, however, defines a port interface "DiagnosticDoIPTriggerVehicleAnnouncementInterface" [SWS_DM_00820], which implies, that an AA could *explicitly* trigger the sending of the "Vehicle Announcement Message". This is not foreseen! An

AA may *indirectly* command the "Vehicle Announcement Message", using the "DiagnosticDoIPActivationLineInterface" [SWS_DM_00830], but not *explicit*.

[SWS_DM_02005] Providing VIN if interface is not available

Upstream requirements: RS_Diag_00026

[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']

[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], DM shall send 6-byte EID value equal '0xFF 0xFF 0xFF 0xFF 0xFF 0xFF'.]

[SWS_DM_02006] Providing EID using manually configured value

Upstream requirements: RS_Diag_00026

[If `DoIpNetworkConfiguration.eidRetrieval` is configured with `DoIpEidRetrievalEnum.eidUseConfigValue`, DM shall obtain the EID as configured in `DoIpFunctionalClusterDesign.eid`.]

[SWS_DM_02007] Providing EID using MAC of network interface

Upstream requirements: RS_Diag_00026

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

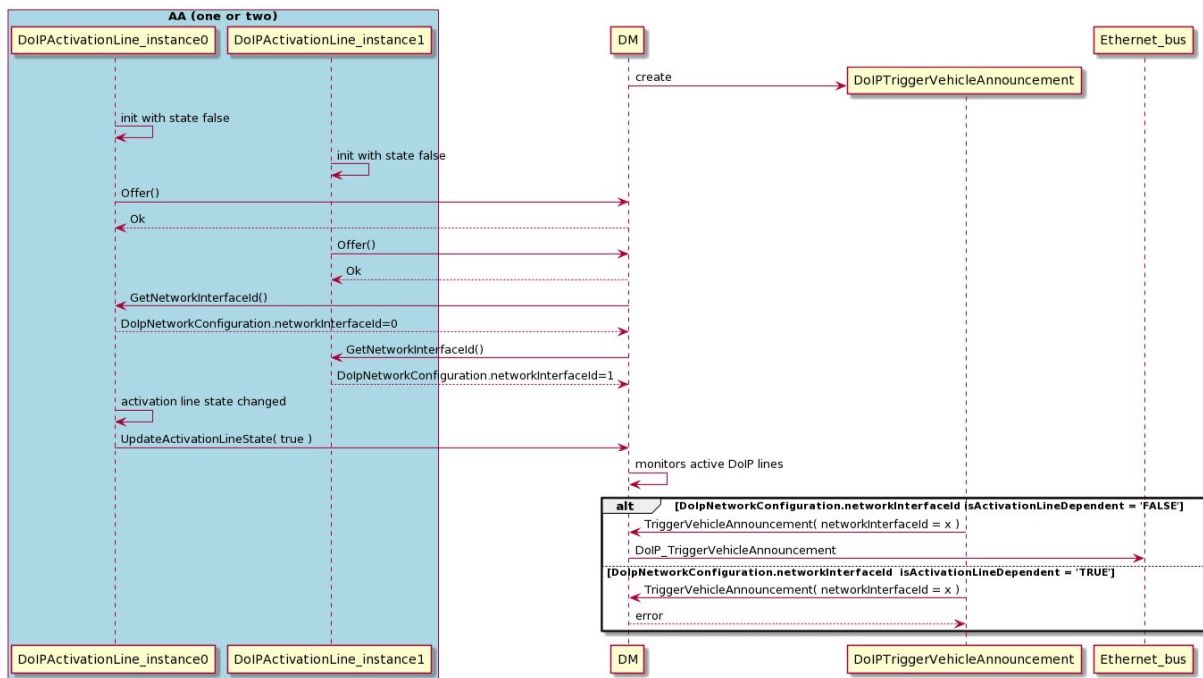


Figure 7.2: Example Sequence Diagram for Activation Line and Vehicle Announcement API use case

[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.secureCom-PropsForTcp](#).]

[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_02106] Valid values for the attribute `maxTesterConnections` of meta-class `DoIpNetworkConfiguration`

Upstream requirements: [RS_Diag_04166](#)

[Values of `DoIpNetworkConfiguration.maxTesterConnections` shall be from '1' to '255'.]

[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 reception 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[5] according to the configuration referenced by [SovdModuleInstantiation.communication-Connector](#).]

[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 m[DNS](#) (Multicast [DNS](#)) as specified by ASAM SOVD[5].]

[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 [23].

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.

7.3.1.1 MetaInfo class

The `ara::diag::MetaInfo` class specifies a mechanism to provide meta information, 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”](#).

Key	Context			String-Format	Description
	<code>kDiagnosticCommunication</code>	<code>kFaultMemory</code>	<code>kSOVD</code>		
<code>kSA</code>	X	-	-	<code>[0-9A-F]{4}</code>	UDS Source Address from which the diagnostic request has been sent. The value is formatted as hexadecimal number. For example tester SA of decimal 240 will have the stringified value "00F0".
<code>kTA</code>	X	-	-	<code>[0-9A-F]{4}</code>	UDS Target Address to which the diagnostic request has been sent. The value is formatted as hexadecimal number. For example TA of decimal 59 will have the stringified value "003B".
<code>kTAType</code>	X	-	-	"PHYS" or "FUNC"	Indicator whether request is functional or physical addressed.
<code>kRequestHandle</code>	X	-	X	<code>[0-9]{1,5}</code>	Key for the RequestHandle, which shall be identical for all API calls within the context of the same diagnostic request. For example for a <code>Validate()</code> and an final <code>Confirmation()</code> call in the context of the same diagnostic request, the same value for this key has to be placed in the metaInfo.
<code>kLocalIP</code>	X	-	-	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 [24]).





kLocalPort	X	-	-	[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	X	-	-	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 [24]).
kRemotePort	X	-	-	[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	-	X	-	[0-9A-F]{6}	DTC number for which this interface is triggered
kBaseUri	-	-	X	URI	Base URI used for the request.
kEntityPath	-	-	X		Entity path used for the request.
kResourcePath	-	-	X		Path to the requested resource or resource collection.
kClientIdentity	-	-	X		Identity of the client. String format is defined by application handling the authentication.

Table 7.1: MetaInfo type definition

For context [kDoIP](#) none of the listed keys is applicable.

7.3.1.1.1 Provision of MetaInfo

[SWS_DM_02094] Provision of source address

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the source address (SA) of the diagnostic client from which the diagnostic request was received within metaInfo using the Key kSA in all the subsequently called `ara::diag` interfaces.]

[SWS_DM_02095] Provision of target address

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the target address (TA) of the diagnostic server to which the diagnostic request was sent within metaInfo using the Key kTA in all the subsequently called `ara::diag` interfaces.]

[SWS_DM_02096] Provision of UDS addressing mode

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the UDS addressing mode (TAType) of the diagnostic request (physical or functional) within metaInfo using the Key kTAType in all the subsequently called `ara::diag` interfaces.]

[SWS_DM_02097] Provision of request handle

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the request handle of the diagnostic request within metaInfo using the Key kRequestHandle in all the subsequently called ara::diag interfaces.]

[SWS_DM_02098] Provision of local IP address

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the local IP address within metaInfo using the Key kLocalIP in all the subsequently called ara::diag interfaces.]

[SWS_DM_02099] Provision of local port number

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the local port number within metaInfo using the Key kLocalPort in all the subsequently called ara::diag interfaces.]

[SWS_DM_02102] Provision of remote IP address

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the remote IP address within metaInfo using the Key kRemoteIP in all the subsequently called ara::diag interfaces.]

[SWS_DM_02103] Provision of remote port number

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the remote port number within metaInfo using the Key kRemotePort in all the subsequently called ara::diag interfaces.]

[SWS_DM_02104] Provision of DTC number

Upstream requirements: [RS_Diag_04170](#)

[The DM shall provide the DTC number for which the interfaces is triggered within metaInfo using the Key kDtc in all the subsequently called ara::diag interfaces for the purpose of extended data and snapshot data collection.]

[SWS_DM_02105] Provision of kDoIP context

Upstream requirements: [RS_Diag_04170](#)

[The DM shall use the context for metaInfo as kDoIP if a DataIdentifier is read according to [\[SWS_DM_00855\]](#).]

7.3.1.1.2 Lifetime of MetalInfo

[SWS_DM_01345] Lifetime of MetalInfo

Upstream requirements: [RS_Diag_04170](#)

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

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

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 $((\text{DiagnosticSecurityAccess.requestSeedId} + 1) / 2)$ 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 re-entrancy 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[5] 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 `<namespace-list-data-identifier>::<data-identifier-interface-name-x>` 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 `<namespace-list-data-identifier>::<data-identifier-interface-name-x>` 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 `<namespace-list-data-identifier>::<data-identifier-interface-name-x>` 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_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 set the `ActivityStatus` to inactive `kInactive`.]

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 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.

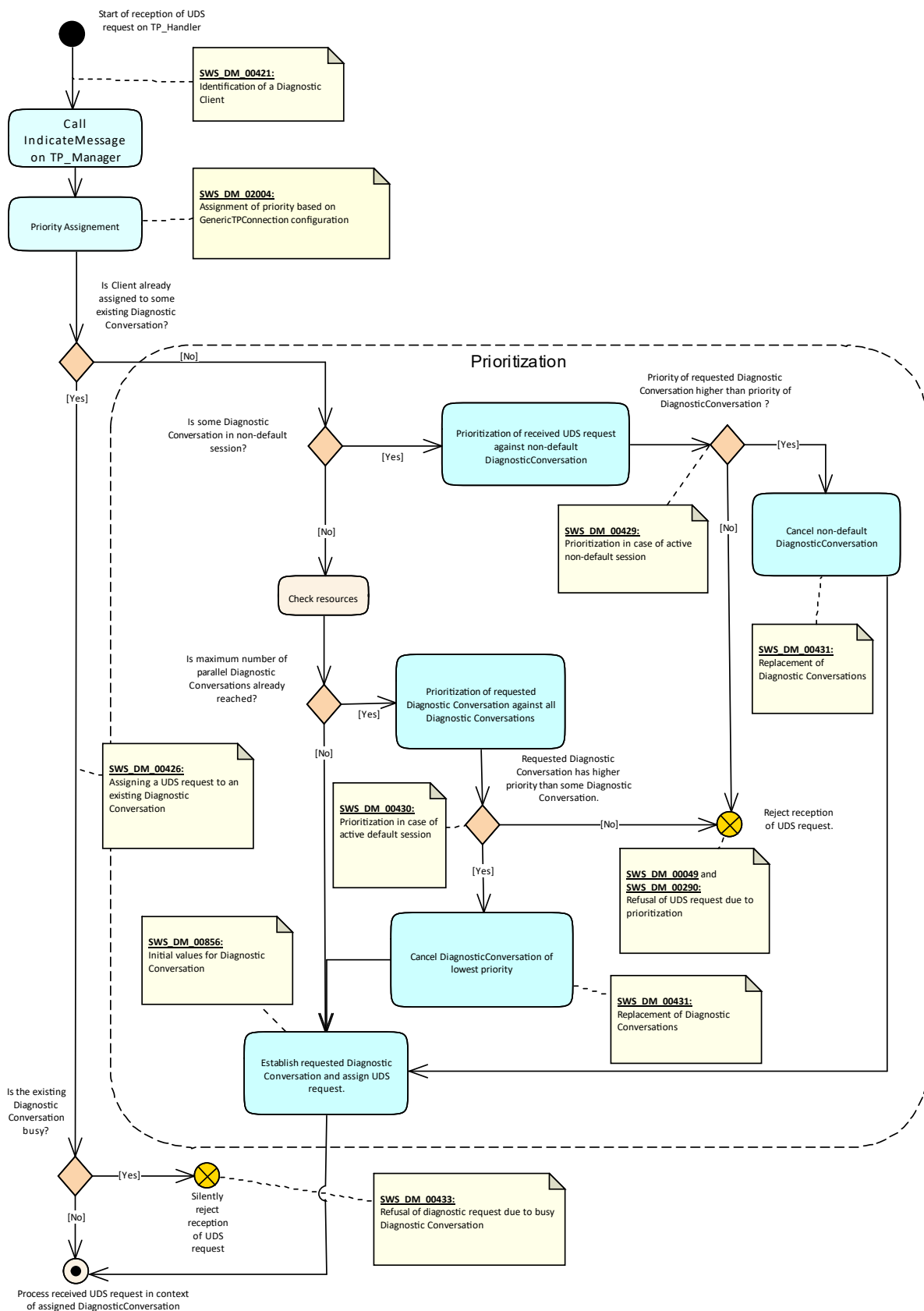


Figure 7.3: UDS request assignment to a Diagnostic Conversation and Prioritization

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 `DoIpNetworkConfigurationDesign.tpConnection.priority` where the source address provided by `apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` is in the range between `DoIpNetworkConfigurationDesign.tpConnection.sourceAddressRangeStart`

and `DoIpNetworkConfigurationDesign.tpConnection.sourceAddressRangeEnd.`]

[SWS_DM_02108] Default value for the attribute `sourceAddressRangeStart` of meta-class `GenericTpConnection`

Upstream requirements: [RS_Diag_04166](#)

[If the attribute `DoIpNetworkConfigurationDesign.tpConnection.sourceAddressRangeStart` is not configured, the `Diagnostic Server instance` shall use a value of 0.]

[SWS_DM_02109] Default value for the attribute `sourceAddressRangeEnd` of meta-class `GenericTpConnection`

Upstream requirements: [RS_Diag_04166](#)

[If the attribute `DoIpNetworkConfigurationDesign.tpConnection.sourceAddressRangeEnd` is not configured, the `Diagnostic Server instance` shall use a value of 65535.]

[SWS_DM_02110] Default value for the attribute `priority` of meta-class `GenericTpConnection`

Upstream requirements: [RS_Diag_04166](#)

[If the attribute `DoIpNetworkConfigurationDesign.tpConnection.priority` is not configured, the `Diagnostic Server instance` shall use a value of 255.]

[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, `DoIpNetworkConfigurationDesign.tpConnection.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 `DoIpNetworkConfigurationDesign.tpConnection.priority` of the requested `Diagnostic Conversation` against the `DoIpNetworkConfigurationDesign.tpConnection.priority` of the given `Diagnostic Conversation` in non-default session. If the `DoIpNetworkConfigurationDesign.tpConnection.priority` of the requested `Diagnostic Conversation` is higher than the `DoIpNetworkConfigurationDesign.tpConnection.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 [DoIp-NetworkConfigurationDesign.tpConnection.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](#)),
- Security Level set to status Locked ([kLocked](#)),
- ActivityStatus to active [kActive](#) and
- ConversationIdentifier matching the value of [apext::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage](#) call.

]

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 AP, 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::DynamicAccessList-DiagServiceBuilder` 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 then 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_04276](#)

[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_04276](#)

[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_04276](#), [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 ([kIncorrectMessageLengthOrInvalidFormat](#)).]

7.3.2.5.2 Supported service checks

[SWS_DM_00099] Supported Service SID level checks

Upstream requirements: [RS_Diag_04276](#)

[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 ([kServiceNotSupported](#)).]

[SWS_DM_00100] Supported Service subfunction level checks

Upstream requirements: [RS_Diag_04276](#)

[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 ([kSubFunctionNotSupported](#)).]

7.3.2.5.3 Session and Security Checks

[SWS_DM_00101] Session Access SID level Permission

Upstream requirements: [RS_Diag_04276](#), [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 identifiers have no access permission in the current session.

the Verification shall be considered as failed and the negative response code shall be 0x7F ([kServiceNotSupportedInActiveSession](#)).]

[SWS_DM_00102] Session Access subfunction level Permission

Upstream requirements: [RS_Diag_04276](#), [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 ([kSubFunctionNotSupportedInActiveSession](#)).]

[SWS_DM_00103] Security Access level Permission

Upstream requirements: [RS_Diag_04276](#), [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 ([kSecurityAccessDenied](#)).]

[SWS_DM_00450] Security Access subfunction level Permission

Upstream requirements: [RS_Diag_04276](#), [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 ([kSecurityAccessDenied](#)).]

[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_04199](#)

[If any [ara::diag::ServiceValidation::Validate](#) returns [kNoProcessing-NoResponse](#), 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 [4]).]

[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](#).]

Atomic conditions in the context of a [DiagnosticEnvConditionFormula](#) are described by means of the abstract meta-class [DiagnosticEnvCompareCondition](#). The class [DiagnosticEnvCompareCondition](#) itself is based on the comparison of a variable at runtime with a constant value.

This variable can be received in two ways:

- Reference to a `DiagnosticDataElement` (when using `DiagnosticEnvDataCondition` subclass of `DiagnosticEnvCompareCondition`)
- Reference to a `PPortPrototype`, typed by a `DiagnosticDataElementInterface` (when using `DiagnosticEnvDataElementCondition` subclass of `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 (kConditionsNotCorrect)`.]

7.3.2.6 UDS response handling

[SWS_DM_01258] Response handling

Upstream requirements: [RS_Diag_04020](#), [RS_Diag_04249](#), [RS_Diag_04277](#)

[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](#), [RS_Diag_04277](#)

[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] Max number of Response Pending Messages

Upstream requirements: [RS_Diag_04016](#), [RS_Diag_04249](#), [RS_Diag_04277](#)

[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](#), [RS_Diag_04277](#)

[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 process 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 pre-defined set of NRC codes that are standardized in ISO 14229-1[\[25\]](#). An application may choose any of the values within [\[SWS_DM_00526\]](#) to trigger the corresponding NRC as result of the service processing. In some rare cases the application will trigger NRC codes that are not standardized in ISO 14229-1[\[25\]](#). These NRCs are in ranges that are or marked as manufacturer specific or reserved. It is explicitly 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 definitions 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 course 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_04277](#)

[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_04277](#)

[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 $S3_{\text{Server}}$ (session timeout) with a server specific value according to [\[SWS_DM_01747\]](#). The timer handling shall be implemented according to ISO 14229-2[\[25\]](#).]

[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 $S3_{\text{Server}}$ timeout. If the parameter `DiagnosticSessionControlClass.s3ServerTimeout` is configured, the [Diagnostic Server instance](#) (Software Cluster) shall take the configured value as the $S3_{\text{Server}}$ 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 ($S3_{\text{Server}}$) 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[\[25\]](#).]

[SWS_DM_00382] Session timeout start

Upstream requirements: [RS_Diag_04006](#), [RS_Diag_04249](#)

[The session timeout timer ($S3_{\text{Server}}$) 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](#).

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	7.3.2.8.6
0x19	ReadDTCInformation	Internally	7.3.2.8.7
0x22	ReadDataByIdentifier	Internally & Externally	7.3.2.8.8
0x27	SecurityAccess	Internally & Externally	7.3.2.8.9
0x28	CommunicationControl	Externally	7.3.2.8.10
0x29	Authentication	Externally	7.3.2.8.11
0x2A	ReadDataByPeriodicIdentifier	Internally	7.3.2.8.12
0x2C	DynamicallyDefineDataIdentifier	Internally	7.3.2.8.13
0x2E	WriteDataByIdentifier	Externally	7.3.2.8.14
0x31	RoutineControl	Externally	7.3.2.8.15
0x34	RequestDownload	Externally	7.3.2.8.16
0x35	RequestUpload	Externally	7.3.2.8.17
0x36	TransferData	Externally	7.3.2.8.18
0x37	RequestTransferExit	Externally	7.3.2.8.19
0x38	RequestFileTransfer	Externally	7.3.2.8.20
0x3E	TesterPresent	Internally	7.3.2.8.21

SID	Service	Support Type	Reference
0x85	ControlDTCSetting	Internally	7.3.2.8.22
0x86	ResponseOnEvent	Internally	7.3.2.8.23

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 (`kRequestOutOfRange`), 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_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 (kSubFunctionNotSupported)`.]

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 (CP) world and ISO 14229-1[1]. Regarding the Adaptive Platform (AP), 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_04279](#)

[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_04279](#)

[The [Diagnostic Server instance](#) shall send a negative response 0x12 (`kSubFunctionNotSupported`), 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.]

7.3.2.8.5.1 Service 0x11 – ECUReset – RapidPowerShutdown sub function

[SWS_DM_01020] EnableRapidPowerShutdown processing

Upstream requirements: [RS_Diag_04279](#)

[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_04279](#)

[If a positive response for subfunction `EnableRapidPowerShutdown` shall be sent, the DM shall put the configured value of `SoftwareClusterDiagnosticDeploymentProps.powerDownTime` in its positive response.]

[SWS_DM_01021] DisableRapidPowerShutdown processing

Upstream requirements: [RS_Diag_04279](#)

[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 [\[SWS_DM_00100\]](#).]

7.3.2.8.5.2 Service 0x11 – ECUReset – all other sub functions

[SWS_DM_00235] ECUReset service processing

Upstream requirements: [RS_Diag_04279](#)

[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_04279](#)

[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_04279](#)

[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_04279](#)

[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_04279](#)

[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 (`kGeneralReject`), 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_04279](#)

[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_04279](#)

[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_04279](#)

[If UDS requests are blocked due to [\[SWS_DM_01022\]](#) and the `ara::core::Future` returned from the call of `ExecuteReset`, the DM shall accept incoming UDS requests again.]

7.3.2.8.6 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](#)

[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 (kRequestOutOfRange)` if the requested `groupOfDTC` has no matching configured `DTC`

group according to [SWS_DM_00064] or configured DTC by `DiagnosticTrouble-CodeUds.udsDtcValue.`]

[SWS_DM_00113] Positive response for UDS service 0x14

Upstream requirements: RS_Diag_04180

[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 (kGeneralProgrammingFailure)` 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 an 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 (kConditionsNotCorrect)` 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 `clearAllDtc`, 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 (kRequestOutOfRange).`]

[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](#)

[The `Diagnostic Server instance` shall send a positive response for a Clear-DiagnosticInformation 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 (`kConditionsNotCorrect`).]

[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 (`kConditionsNotCorrect`), if all DTCs of that group of DTC forbid the clearance according to [\[SWS_DM_00163\]](#) or [\[SWS_DM_00896\]](#).]

7.3.2.8.6.1 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)`.]

7.3.2.8.7 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](#).]

7.3.2.8.7.1 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 [\[SWS_DM_00062\]](#).]

7.3.2.8.7.2 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'.]

7.3.2.8.7.3 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'.]

7.3.2.8.7.4 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'.]

7.3.2.8.7.5 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'.]

7.3.2.8.7.6 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].]

7.3.2.8.7.7 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.]

7.3.2.8.7.8 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'.]

7.3.2.8.7.9 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'.]

7.3.2.8.7.10 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'.]

7.3.2.8.7.11 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'.]

7.3.2.8.8 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_04275](#)

[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_04275](#), [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 \(kIncorrectMessageLengthOrInvalidFormat\)](#).]

[SWS_DM_00409] Check supported DataIdentifier

Upstream requirements: [RS_Diag_04275](#)

[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_04275](#), [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_04275](#), [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 [DiagnosticServer 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 [\[SWS_DM_00393\]](#)

]

Access of single data element(s)

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.

Multi-DID-Requests

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.

Refer to ISO 14229-1[1]

7.3.2.8.9 Service 0x27 – SecurityAccess

[SWS_DM_00236] Realization of UDS service 0x27 SecurityAccess

Upstream requirements: [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_04005](#)

[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_04005](#), [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 \(kIncorrectMessageLengthOrInvalidFormat\)](#).]

[SWS_DM_00864] Checking Supported Subfunction for CompareKey

Upstream requirements: [RS_Diag_04005](#)

[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_04005](#)

[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 ([kSubFunctionNotSupported](#)) 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 [\[SWS_DM_00272\]](#).

7.3.2.8.10 Service 0x28 – CommunicationControl

[SWS_DM_00140] Realisation of UDS service 0x28 CommunicationControl

Upstream requirements: [RS_Diag_04278](#)

[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_04278](#)

[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 ([kSubFunctionNotSupported](#)) 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_04278](#)

[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_04278](#)

[If the external processor did raise no [ApApplicationError](#), the [Diagnostic Server instance](#) shall return a positive response.]

7.3.2.8.11 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.

0x00 deAuthenticate

0x01 verifyCertificateUnidirectional

0x02 verifyCertificateBidirectional

0x03 proofOfOwnership

0x04 transmitCertificate

0x08 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

]

7.3.2.8.11.1 Service 0x29 – deAuthenticate (0x00)

[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.]

7.3.2.8.11.2 Service 0x29 – verifyCertificateUnidirectional (0x01)

[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.]

Handling of multiple verifyCertificateUnidirectional requests

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.

7.3.2.8.11.3 Service 0x29 – verifyCertificateBidirectional (0x02)**[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 (LOCSE) and lengthOfProofOfOwnershipServer (LPOWSE) 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 (POWSE) 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.]

Handling of multiple verifyCertificateBidirectional requests

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` sub-function request

7.3.2.8.11.4 Service 0x29 – proofOfOwnership (0x03)**[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` (POWNC) 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.

7.3.2.8.11.5 Service 0x29 – transmitCertificate (0x04)**[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 or 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 authentication-ReturnParameter (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.]

7.3.2.8.11.6 Service 0x29 – authenticationConfiguration (0x08)**[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.]

7.3.2.8.12 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 ([kIncorrectMessageLengthOrInvalidFormat](#)).]

[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 ([kIncorrectMessageLengthOrInvalidFormat](#)).]

[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 ([kRequestOutOfRange](#)).]

[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 ([kRequestOutOfRange](#)).]

[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([kRequestOutOfRange](#)).]

[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(**kRequestOutOfRange**).]

[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 DataIdentifier's 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 ([kRequestOutOfRange](#)).]

[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 PIDID transmission is triggered for a requested transmission rate, the DM shall transmit all PDIDs on the periodic transmission by calling `apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit` on the periodic connection starting with the first PIDID 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.]

7.3.2.8.13 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 [DDID](#). In default session the last defined [DDID](#) will be used. The client can switch to a non-default session ensure that no other testers interfere with it's [DDIDs](#).

[SWS_DM_01071] No persistency of defined [DDIDs](#)

Upstream requirements: [RS_Diag_04246](#)

[If [DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling](#) is set to volatile, the DM shall initialize all [DDIDs](#) as not present at DM Initialization.]

[SWS_DM_01072] Persistency of defined [DDIDs](#)

Upstream requirements: [RS_Diag_04246](#)

[If [DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling](#) is set to nonVolatile, the DM shall persist [DDIDs](#) configured by the UDS service 0x2C and restore the [DDIDs](#) 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 [DDID](#) 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 (**kRequestOutOfRange**).]

[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 (**kRequestOutOfRange**).]

[SWS_DM_01076] Clearing all configured **DDIDs**

Upstream requirements: [RS_Diag_04246](#)

[On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clear-DynamicallyDefinedDataIdentifier and no dynamicallyDefinedDataIdentifier in the request, the DM shall clear all the configured **DDIDs**.]

[SWS_DM_01077] Clearing individual configured **DDIDs**

Upstream requirements: [RS_Diag_04246](#)

[On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clear-DynamicallyDefinedDataIdentifier and one dynamicallyDefinedDataIdentifier in the request, the DM shall clear the according **DDID** from the request.]

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

[SWS_DM_01078] Clear **DDIDs** 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 **DDIDs** that have sourceDataIdentifiers that are no longer supported in the new session and security level.]

[SWS_DM_01079] Session check for **DDID**

Upstream requirements: [RS_Diag_04246](#), [RS_Diag_04226](#)

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

[SWS_DM_01080] Security level check for [DDID](#)

Upstream requirements: [RS_Diag_04246](#), [RS_Diag_04227](#)

[On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the [DDIDs](#) can be defined in the security level according to [\[SWS_DM_00414\]](#). If the [DDID](#) 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 (`kRequestOutOfRange`).]

[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 [DDID](#) is requested with UDS service 0x22 or 0x2A.

[SWS_DM_01083] Use of configured [DID](#) ports to get [DDID](#) data

Upstream requirements: [RS_Diag_04246](#)

[If a [DDID](#) read operation is requested, the DM shall retrieve the data for the `dataRecord` of the [DDID](#) by using the configuration of the contained [DIDs](#).]

7.3.2.8.14 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_04280](#)

[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_04280](#)

[On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier ([DID](#)) 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_04280](#), [RS_Diag_04226](#)

[On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier ([DID](#)) 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_04280](#), [RS_Diag_04227](#)

[On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier ([DID](#)) 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 [RPort-Prototypes](#).]

7.3.2.8.15 Service 0x31 – RoutineControl**[SWS_DM_00201] Realization of UDS service RoutineControl (0x31)**

Upstream requirements: [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_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 ([kRequestOutOfRange](#)) shall be returned.]

[SWS_DM_00448] Check supported RoutineIdentifier in active session

Upstream requirements: [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_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_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 ([kSubFunctionNotSupported](#)) shall be returned.]

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

Upstream requirements: [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_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_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_04224](#)

[The `Diagnostic Server` instance shall utilize the associated `RPortPrototype` if it is typed by the `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>` class and call its `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>::Start` function to process the subfunction `startRoutine`.]

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

Upstream requirements: [RS_Diag_04224](#)

[The `Diagnostic Server` instance shall utilize the associated `RPortPrototype` if it is typed by the `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>` class and call its `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>::Stop` function to process the subfunction `stopRoutine`.]

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

Upstream requirements: [RS_Diag_04224](#)

[The `Diagnostic Server` instance shall utilize the associated `RPortPrototype` if it is typed by the `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>` class and call its `<routine-interface-hierarchical-namespace-list>::<routine-interface-name-x>::RequestResult` function to process the subfunction `requestRoutineResults`.]

7.3.2.8.16 Service 0x34 – RequestDownload

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

Upstream requirements: [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_04033](#)

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

7.3.2.8.17 Service 0x35 – RequestUpload

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

Upstream requirements: [RS_Diag_04033](#)

[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).]

7.3.2.8.18 Service 0x36 – TransferData

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

Upstream requirements: [RS_Diag_04033](#)

[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 [\[SWS_DM_01096\]](#).

7.3.2.8.19 Service 0x37 – RequestTransferExit

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

Upstream requirements: [RS_Diag_04033](#)

[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 [\[SWS_DM_01097\]](#).

7.3.2.8.20 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 individual 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](#).

7.3.2.8.21 Service 0x3E – TesterPresent

[SWS_DM_00126] Realisation of UDS service 0x3E TesterPresent

Upstream requirements: [RS_Diag_04198](#)

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

7.3.2.8.22 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.

The ControlDTCSetting service in ISO 14229-1[1] allows the optional parameter DTCSettingControlOptionRecord. The [Diagnostic Server instance](#) always supports this parameter. This effectively means that a ControlDTCSetting service with or without DTCSettingControlOptionRecord is processed in the same way. As defined in [SWS_DM_00231] the DTCSettingControlOptionRecord is limited to the one and only 0xFFFFFFFF value.

[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_04115](#)

[If the [Diagnostic Server instance](#) receives a UDS service ControlDTCSetting (0x85) request with DTCSettingControlOptionRecord != 0xFFFFFFFF, the [Diagnostic Server instance](#) shall send a **NRC** 0x31 ([kRequestOutOfRange](#)).]

[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 0xFFFFFFFF, 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 0xFFFFFFFF, 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.]

7.3.2.8.23 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)

Upstream requirements: [RS_Diag_04160](#)

[

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 DiagnosticResponseOnEvent-Class.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.maxNumChangeOfDataIdentifierEvents

Upstream requirements: [RS_Diag_04160](#)

[The [Diagnostic Server instance](#) shall map the DEXT parameter [DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents](#) to the ISO 14229-1[1] parameter [maxNumChangeOfDataIdentifierEvents](#).]

[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 functionality of the RoE service with the storeState bit (Bit6) set to true according to ISO 14229-1[1].]

7.3.2.8.24 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 [[23](#)].

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-reentrant implementation, the DM shall return the error for "Conflicted state" as per [\[5\]](#).]

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 – Locks**[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 SOVD – Validation of Environmental Conditions

[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 [\[SWS_DM_00111\]](#), [\[SWS_DM_00112\]](#), [\[SWS_DM_00286\]](#) and [\[SWS_DM_00287\]](#), using [DiagnosticEnvSovdDataCondition](#) for SOVD.]

[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 SOVD – Service Validation of 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 Serialization & 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 14229-1 [1]) the data is expressed by internal data types. The data is defined by `DiagnosticSovdContentElement` and aggregated by `DiagnosticSovdContent`. To express this transformation `SwDataDefProps` are aggregated by `DiagnosticSovdContentElement`.

This section describes how these transformations shall be interpreted for SOVD and how the subset of a JSON schema that represents a `DiagnosticSovdContent` shall be derived based on Diagnostic Extract Template[4], so that the `Diagnostic Server instance` can offer consistent ASAM SOVD [5] [14] [13] APIs.

7.3.3.3.1 SOVD Data – Serialization

SOVD data is serialized, using settings from `DiagnosticSovdProps.singleElementSerialization`. The setting object is supported by this version of the standard. See also chapter Common primitive data types in ISO 17978-3(2026) [13].

7.3.3.3.2 SOVD Data – Conversion

[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 `DiagnosticSovdContentElement` with `DiagnosticSovdPrimitiveContentElement.swDataDefProps` 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[4] 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 `DiagnosticSovdPrimitiveContentElement.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 [DiagnosticSovdContentElement](#) with [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#) with [baseTypeEncoding](#) NONE or 2C and a [compuMethod](#) of category TEXTTABLE as defined in Diagnostic Extract Template[4] 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](#) of the corresponding [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#).]

[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 [DiagnosticSovdContentElement](#) with [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#) with [baseTypeEncoding](#) NONE or 2C and a [compuMethod](#) of category SCALE_LINEAR_AND_TEXTTABLE, SCALE_RAT_AND_TEXTTABLE as defined in Diagnostic Extract Template[4] 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](#) of the corresponding [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#).]

[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 [DiagnosticSovdContentElement](#) with [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#) with [baseTypeEncoding](#) NONE or 2C and a [compuMethod](#) of category TAB_NOINTP as defined in Diagnostic Extract Template[4] 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](#) of the corresponding [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#).]

[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 [DiagnosticSovdContentElement](#) with [DiagnosticSovdPrimitiveContentElement.swDataDefProps](#) with [baseTypeEncoding](#) NONE or 2C and a [compuMethod](#) of category BITFIELD_TEXTTABLE as defined in Diagnostic Extract Template[4] 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 `DiagnosticSovdContentElement` with `DiagnosticSovdPrimitiveContentElement.swDataDefProps` with `baseTypeEncoding` IEEE754 as defined in Diagnostic Extract Template[4] 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 `DiagnosticSovdContentElement` with `DiagnosticSovdPrimitiveContentElement.swDataDefProps` with `baseTypeEncoding` of ISO-8859-1, ISO-8859-2, WINDOWS-1252, UTF-8, UTF-16 or UCS-2 as defined in Diagnostic Extract Template[4] 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[5][14][13].]

[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 `DiagnosticSovdContentElement` with `DiagnosticSovdPrimitiveContentElement.swDataDefProps` with a unit referenced as defined in Diagnostic Extract Template[4] 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 `DiagnosticSovdContentElement` with `DiagnosticSovdArrayContentElement` shall be of type `array` with content type derived according to `DiagnosticSovdPrimitiveContentElement.swDataDefProps`.

The JSON property `maxItems` shall be derived from `DiagnosticSovdArrayContentElement.maxNumberOfElements` of the `DiagnosticSovdArrayContentElement`.]

7.3.3.4 Standardized SOVD APIs

7.3.3.4.1 SOVD resource 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[5].]

[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 SOVD resource 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[5].]

[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 SOVD data-categories

Data that can be read or written by SOVD can be grouped into various data-categories. The SOVD request to retrieve data-categories is used by the SOVD client to obtain a list of supported data-categories, supported by the SOVD entity.

Available standard SOVD data-categories used within data resource collections are identifications, measurements, parameters and system information. Additional custom categories are also possible and can be used.

The definition of data-categories is derived by the [DiagnosticSovdDataCategory.category](#):

Each data-category that occurs at least once within a [DiagnosticSovdDataCategory.category](#) referenced from a [DiagnosticSovdData](#), represents a supported data-category.

[SWS_DM_01449] Support of SOVD method Retrieve Categories Supported by a Data Resource Collection

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[The DM shall support the SOVD method `Retrieve Categories Supported by a Data Resource Collection` according to ASAM SOVD[5][14][13].]

[SWS_DM_01448] Retrieve resource Data categories

Upstream requirements: [RS_Diag_04260](#)

[If the SOVD method `Retrieve Categories Supported by a Data Resource Collection` is received, the DM shall send a response body according to ASAM SOVD[5][14][13] and add for each available [DiagnosticSovdDataCategory.category](#) one element with the attribute `item` set to [DiagnosticSovdDataCategory.category](#) to the attribute `items` of the SOVD `DataCategoryInformation` array.]

Support of optional attribute `category_translation_id`

The optional ASAM SOVD[5][14][13] attribute `category_translation_id` is not supported by AUTOSAR, as this information is not available in the manifest.

7.3.3.4.4 SOVD data-groups

SOVD data-groups are used to structure data in a specific way (e.g., engine performance related, identification data ...) and are associated with a category. The category itself is defined by the `shortName` of `DiagnosticSovdGroup`.

[SWS_DM_01446] Support of SOVD method Retrieve Groups Supported by a Data Resource Collection

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[The DM shall support the SOVD method Retrieve Groups Supported by a Data Resource Collection according to ASAM SOVD[5][14].]

[SWS_DM_02160] Retrieve resource Data groups

Upstream requirements: [RS_Diag_04260](#)

[If the SOVD method Retrieve Groups Supported by a Data Resource Collection is received, the DM shall send a response body according to ASAM SOVD[5][14][13] and add for each available `DiagnosticSovdGroup` referenced from a `DiagnosticSovdDataCategory` one `ValueGroup` item with attribute `id` set to the `shortName` of the `DiagnosticSovdGroup` and `category` set to `DiagnosticSovdDataCategory.category`.]

Support of optional attributes `category_translation_id`, `group`, `group_translation_id`

The optional ASAM SOVD[5][14][13] attributes `category_translation_id`, `group` and `group_translation_id` are not supported by AUTOSAR, as this information is not available in the manifest.

7.3.3.4.5 SOVD method Query entity data

This chapter describes the capability of the SOVD entity to provide a list of all available data resources. The returned data may be filtered by their data-category or data-group.

[SWS_DM_01442] Support of SOVD method Query for data from an Entity

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[The DM shall support the SOVD method Query for data from an Entity according to ASAMSOVD[5][14].]

[SWS_DM_02161] Retrieval of SOVD entity data

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[If the SOVD method Query for data from an Entity is received without parameter, the DM shall send a response body according to ASAMSOVD[5][14] and add

for each available `DiagnosticSovdData` an element to the `items` array with the attributes (SOVD datatype `ValueMetaData`):

`id` set to the `shortName` of the `DiagnosticSovdContent`,
`name` set to the `longName` of the `DiagnosticSovdContent`,
`translation_id` (optional SOVD parameter, not yet supported by AUTOSAR),
`category` set to `DiagnosticSovdDataCategory.category` of the `DiagnosticSovdDataCategory` referenced by `DiagnosticSovdData.sovdDataCategory`,
`group` (if configured) set to the `shortName` of the `DiagnosticSovdGroup`.
]

[SWS_DM_02162] SOVD method Query for data from an Entity with categories filter

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[If the SOVD method `Query` for data from an Entity is received with a `categories` query parameter, the DM shall send a response body according to [\[SWS_DM_02161\]](#), but only include the `elements`, where the filter from the request matches the `DiagnosticSovdDataCategory.category`.]

[SWS_DM_02163] SOVD method Query for data from an Entity with group filter

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[If the SOVD method `Query` for data from an Entity is received with a `groups` query parameter, the DM shall send a response body according to [\[SWS_DM_02161\]](#), but only include the `elements`, where the filter from the request matches the `shortName` of the `DiagnosticSovdGroup`.]

7.3.3.4.6 SOVD method 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[5].]

7.3.3.4.7 SOVD method 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[5].]

7.3.3.5 Configurable SOVD APIs

This section describes how the [Diagnostic Server instance](#) realizes the configurable SOVD APIs as specified in ASAM SOVD[5]. By intent a dedicated set of APIs is used for SOVD.

Despite there are already APIs for UDS for similar use cases, such as "UDS DataIdentifier vs. SOVD Data" or "UDS Routine vs. SOVD operation", there are slightly different concepts in SOVD, starting with granularity of authentication or operation timeouts that are better covered with dedicated SOVD APIs.

These SOVD methods are realized explicitly:

- SOVD data
- SOVD faults
- SOVD operations
- SOVD bulk-data
- SOVD configuration as parameter and configuration as bulk-data
- SOVD software update
- SOVD logs

7.3.3.5.1 SOVD API – Data Access

An SOVD `entity` allows read and write access to various types of data.

[SWS_DM_01433] Support of SOVD method Read a data resource from an Entity

Upstream requirements: [RS_Diag_04260](#)

[For each [DiagnosticSovdData](#), the DMshall provide the SOVD method Read a data resource from an Entity according to ASAM SOVD [5][14].]

Definition of SOVD data

SOVD data is defined by the `DiagnosticSovdContent`, which itself, is composed out of one or more `DiagnosticSovdContentElement`(s). The DM uses the interface `ara::diag::GenericDiagnosticSovdContent` to read or write the entire data, while applying the data conversion rules defined in [Chapter 7.3.3.3.2 “SOVD Data – Conversion”](#).

[SWS_DM_01432] SOVD method Read a data resource from an Entity – data query

Upstream requirements: [RS_Diag_04260](#)

[If the SOVD method `Read a data resource` is received, the DM shall call the corresponding `ara::diag::GenericDiagnosticSovdContent::Read` method and send a response body according to ASAM SOVD [5][14] and set the following attribute values:

id to `shortName` of `DiagnosticSovdContent`,

data as enumeration of all `DiagnosticSovdContentElement`s aggregated by the `DiagnosticSovdContent` with:

ValueName set to the `shortName` of the `DiagnosticSovdContentElement`

Value retrieved by [\[SWS_DM_02165\]](#)

]

[SWS_DM_02164] Retrieval of SOVD content

Upstream requirements: [RS_Diag_04260](#)

[To read SOVD data, the DM shall use the `ara::diag::GenericDiagnosticSovdContent::Read` method that is mapped via `DiagnosticSovdContent-PortMapping`.]

[SWS_DM_02165] Retrieval of SOVD content element data

Upstream requirements: [RS_Diag_04260](#)

[To retrieve the value of an `DiagnosticSovdContentElement`, the DM shall evaluate the `DiagnosticSovdContentElementData` provided by [\[SWS_DM_02164\]](#) and retrieve the `data` of the requested `DiagnosticSovdContentElement`(s) from the returned `DiagnosticSovdContentElementData` object, by fetching the content via `ara::diag::DiagnosticSovdContentElementDataWithKey::GetContent`, while `ara::diag::DiagnosticSovdContentElementDataWithKey::GetName` is matching the `shortName` of the `DiagnosticSovdContentElement`.]

[SWS_DM_01431] Support of SOVD method Write a data resource from an Entity*Upstream requirements:* [RS_Diag_04260](#)

[For each [DiagnosticSovdData](#), the DM shall provide the SOVD method Write a data resource from an Entity according to ASAM SOVD [5][14].

The write method is optional and is based on the presence of [DiagnosticSovdData](#).
[DiagnosticSovdMethod.put.](#)]

[SWS_DM_02166] SOVD method Write a data resource from an Entity – data query*Upstream requirements:* [RS_Diag_04260](#)

[If the SOVD method Write a data resource is received, the DM shall call the corresponding [GenericDiagnosticSovdContent::Write](#) method with the [DiagnosticSovdContentData](#) data created with:

- [DiagnosticSovdContentElementDataWithKey::SetName](#) with the [shortName](#) of the [DiagnosticSovdContentElement](#)
- [DiagnosticSovdContentElementDataWithKey::SetContent](#) with the value derived from the conversation rules defined in [Chapter 7.3.3.3.2 “SOVD Data – Conversion”](#).

]

7.3.3.5.2 SOVD API – Configuration

The ASAM SOVD[5] 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 [DiagnosticSovdContent](#) under usage of [DiagnosticSovdConfigContentMapping](#).
- Configurations as parameter or Configurations as bulk-data realized explicitly 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[5].]

7.3.3.5.2.1 Configuration – 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 explicitly 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 explicitly modelled SOVD Configuration the [Diagnostic Server instance](#) shall call [ara::diag::SovdConfiguration::RequestPutConfiguration](#).]

7.3.3.5.3 SOVD API – Data Lists**[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[5].]

[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 SOVD API – 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_02120] Response Body for SOVD fault attributes

Upstream requirements: [RS_Diag_04259](#)

[The DM shall map the following data to a SOVD fault attributes response body as listed in table [\[SWS_DM_01567\]](#).]

[SWS_DM_01567] Response Body for SOVD fault attributes table

Upstream requirements: [RS_Diag_04259](#)

[

SOVD attribute	Deviation rules
<code>code</code>	<code>udsDtcValue</code> of DiagnosticTroubleCodeUds
<code>scope</code>	<code>shortName</code> of corresponding DiagnosticMemoryDestination
<code>display_code</code>	<code>shortName</code> of DiagnosticTroubleCodeUds
<code>fault_name</code>	<code>longName</code> of DiagnosticTroubleCodeUds
<code>severity</code>	<code>severity</code> 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_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_02119] Response Body for SOVD fault environment_data

Upstream requirements: [RS_Diag_04259](#)

[The DM shall map the following data to a SOVD fault `environment_data` table response body, as listed in table [\[SWS_DM_01561\]](#).]

[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
<code>environment_data</code>	object	Available if snapshot record data or extended data is supported for the DTC.
<code>environment_data / extended_data_records</code>	object	Available if at least one DiagnosticExtendedDataRecord is configured for the DTC.

▽



environment_data / extended_data_records / <extended data record content>	object	<extended data record content> is a list of all <code>DiagnosticExtendedDataRecords</code> that are referred via <code>DiagnosticTroubleCodeProps.extendedDataRecord</code> . For each <code>DiagnosticExtendedDataRecords</code> the <code>DiagnosticExtendedDataRecord.shortName</code> shall be used for the name and the captured value read in [SWS_DM_00895] and [SWS_DM_00949]. The content shall be represented analog to the content of the SOVD method <code>Read</code> single data value from an Entity.
environment_data / snapshots	array	Available if snapshot record data is <code>configured</code> for the DTC.
environment_data / snapshots / {Array Element}	object	Each <code>freeze frame</code> shall be represented by one <code>element</code> in the array.
environment_data / snapshots / {Array Element} / name	string	In case <code>DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration</code> is set to <code>configured</code> the attribute <code>name</code> shall be derived from the <code>shortName</code> of the corresponding <code>DiagnosticFreezeFrame</code> . In case <code>DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration</code> is set to <code>calculated</code> the attribute <code>name</code> shall be <code>calculated</code> following the pattern "occurrence_{occ}", where "{occ}" represents the record numeration. The first occurrence shall always use the fix string "first_occurrence". The last occurrence shall always use the fix string "last_occurrence".





<pre>environment_data / snapshots / {Array Element} / data / { DiagnosticDataElement. shortName} + {Diagnos- ticSovdContentElement. shortName}</pre>	Snapshot data consists of captured snapshot data from the UDS and SOVD modelling and both data sources are combined here. For UDS the key <code>DiagnosticDataElement.shortName</code> shall be derived from the <code>shortName</code> of the corresponding <code>DiagnosticDataElement</code>. The object shall contain the Diagnostic Snapshot Record of the corresponding fault. For SOVD the key <code>DiagnosticSovdContentElement.shortName</code> shall be derived from the <code>shortName</code> of the corresponding <code>DiagnosticSovdContentElement</code>. The object shall contain the Diagnostic Snapshot Record of the corresponding fault. The UDS and SOVD based data 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 All Faults 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 Single Fault 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

Upstream requirements: [RS_Diag_04259](#)

[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

Upstream requirements: [RS_Diag_04259](#)

[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 SOVD API – Operations**[SWS_DM_02178] Support of SOVD method control of operations from an Entity**

Upstream requirements: [RS_Diag_04258](#), [RS_Diag_04260](#)

[The DM shall support the SOVD method control of operations according to ASAM SOVD[5][14].]

[SWS_DM_01405] SOVD operations

Upstream requirements: [RS_Diag_04262](#)

[Each [DiagnosticSovdOperation](#) shall express a SOVD operation.]

[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, (see [7.3.3.5.5.5](#))
- Terminate execution of an operation, (see [7.3.3.5.5.6](#))
- Read the status of an operation execution (see [7.3.3.5.5.7](#))

shall be represented as JSON object literal.

The literal shall have one entry per [element](#) with key derived from the [shortName](#) of the corresponding [DiagnosticSovdContentElement](#).]

7.3.3.5.5.1 Operations – Query for operations from an Entity

[SWS_DM_01404] Support of SOVD method Query for operations from an Entity

Upstream requirements: [RS_Diag_04262](#), [RS_Diag_04258](#)

[For each [DiagnosticSovdOperation](#), the [Diagnostic Server instance](#) shall provide the SOVD method Query for operations from an Entity according to ASAM SOVD[5][14].]

[SWS_DM_02167] Retrieval of SOVD entity operations

Upstream requirements: [RS_Diag_04262](#), [RS_Diag_04258](#)

[If the SOVD method Query for operations from an Entity is received, the DM shall send a response body according to ASAM SOVD[5][14] and add for each available [DiagnosticSovdOperation](#) a SOVD [OperationDescription](#) item with the attributes:

id set to the [shortName](#) of the [DiagnosticSovdOperation](#) [SWS_DM_01402],
name set to the [longName](#) of the [DiagnosticSovdOperation](#) [SWS_DM_01401],
proximity_proof_required set to [DiagnosticSovdOperation.proximityProofRequired](#) [SWS_DM_01400],
asynchronous_execution set to FALSE, if [DiagnosticSovdOperationPrimitive](#) only aggregates a [DiagnosticSovdOperationStart](#) ([DiagnosticSovdOperation.start](#)), otherwise it is set to TRUE [SWS_DM_01399].

]

[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 [DiagnosticSovdOperation](#).]

[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 [DiagnosticSovdOperation](#).]

[SWS_DM_01400] SOVD operation attribute proximity_proof_required

Upstream requirements: [RS_Diag_04262](#)

[The SOVD attribute [proximity_proof_required](#) of a SOVD operation shall be derived from the [proximityProofRequired](#) of the corresponding [DiagnosticSovdOperation](#).]

[SWS_DM_01399] SOVD operation attribute `asynchronous_execution`

Upstream requirements: [RS_Diag_04262](#)

[If the `DiagnosticSovdOperationPrimitive` only aggregates a `DiagnosticSovdOperationStart` (`DiagnosticSovdOperation.start`), the attribute `asynchronous_execution` of the corresponding SOVD operation shall be `FALSE`, else the attribute shall be `TRUE`.]

7.3.3.5.5.2 Operations – Query for executions of an operation

The SOVD server allows multiple parallel executed operations. Each operation is assigned to a unique identifier at runtime. The query for executions of an operation returns active execution identifiers to the SOVD client.

[SWS_DM_01394] Support of SOVD method Query for executions of an operation

Upstream requirements: [RS_Diag_04260](#), [RS_Diag_04258](#)

[For each SOVD operation supporting `asynchronous_execution`, the `Diagnostic Server instance` shall support the SOVD method Query for executions of an operation according to ASAM SOVD[5][14].]

[SWS_DM_02170] Retrieval of SOVD entity executions of an operation

Upstream requirements: [RS_Diag_04262](#), [RS_Diag_04258](#)

[If the SOVD method Query for executions of an operation is received, the DM shall send a response body according to ASAM SOVD[5][14] and add for each available `DiagnosticSovdOperation`, a SOVD `OperationDescription` item string with the dynamically assigned `execution-id` from [\[SWS_DM_02177\]](#) to that operation.]

7.3.3.5.5.3 Operations – Proximity Management

The ASAM SOVD standard[5][14] uses the `proximity_proof` method to ensure, that certain operations are executed only, in case a supervising person (e.g. technician) is physically present at the vehicle, while an operation is executed.

These operations have `DiagnosticSovdOperation.proximityProofRequired` set to `TRUE`.

To realize the `proximity_proof`, the interface `ara::diag::SovdProximityChallenge` is provided. The intention of the interface is to provide a `proximity_challenge` for all clients and all operations. A client itself can validate itself by knowing the "answer" on the `proximity_challenge` and therefore proof, that there is a supervising person present.

It is assumed, that there is only a single `proximity_challenge` present at any point in time, that is valid for all SOVD clients and all operations. Therefore, a typical application implementing `ara::diag::SovdProximityChallenge` does not need to manage an individual `proximity_challenge` for each requested operation and for each client.

The lifetime of a `proximity_challenge` is typically set to a fixed time by the application (e.g. 2 minutes).

If further complexity, such as different `proximity_challenges` per client are needed, the `ara::diag::MetaInfo` object can be evaluated by the application, to implement client or operation specific `proximity_challenges`. But in most cases this will not be necessary.

[SWS_DM_01398] Providing the SOVD operation `proximity_challenge`

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD operation has `DiagnosticSovdOperation.proximityProofRequired` set to `TRUE`, in each response body where the attribute `proximity_challenge` is required, the `Diagnostic Server instance` shall return the attribute in the response body of the SOVD method with the value returned by `ara::diag::SovdProximityChallenge::GetChallenge`.]

[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_01395] Validation of SOVD operation `proximity_response`

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD request body contains a `proximity_response`, the `Diagnostic Server instance` shall call the method `ara::diag::SovdProximityChallenge::ValidateResponse` with the parameter `proximityResponse` set to the content from the request body and consider the `proximity_challenge` valid, if `kProximityChallengeValidationSuccessful` is returned.]

[SWS_DM_02171] Operation execution without required `proximity_challenge`

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method to execute an operation is received and the `DiagnosticSovdOperation.proximityProofRequired` is set to `TRUE` and the SOVD request body does not contain a `proximity_response`, the DM shall deny the request and return an SOVD error 400 (Bad Request).]

[SWS_DM_02172] Operation execution with wrong proximity_challenge

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method to execute an operation is received and the [DiagnosticSovdOperation.proximityProofRequired](#) is set to TRUE for this operation and the validation of the `proximity_response` according to [\[SWS_DM_01395\]](#) fails with [kProximityChallengeValidationFailed](#), the DM shall deny the request and return an SOVD error 400 (Bad Request).]

[SWS_DM_02173] Operation execution with successful proximity_challenge

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method to execute an operation is received and the [DiagnosticSovdOperation.proximityProofRequired](#) is set to TRUE for this operation, the DM shall only process the execution request, if the validation of the `proximity_response` according to [\[SWS_DM_01395\]](#) succeeds with [kProximityChallengeValidationSuccessful](#).]

7.3.3.5.4 Operations – Retrieve details of an operation

[SWS_DM_01403] Support of SOVD method Retrieve details of an operation

Upstream requirements: [RS_Diag_04262](#), [RS_Diag_04258](#)

[For each [DiagnosticSovdOperation](#), the [Diagnostic Server](#) instance shall provide the SOVD method `Query` for operations from an Entity according to ASAM SOVD[5][14].]

[SWS_DM_02175] Retrieval of SOVD details of an operation

Upstream requirements: [RS_Diag_04262](#)

[If the SOVD method `details of an operation` is received, the DM shall send a response body according to ASAM SOVD[5][14] and add for the requested [DiagnosticSovdOperation.shortName](#):

OperationDescription with the attributes as defined in [\[SWS_DM_02167\]](#)

proximity_challenge (optional) as defined in [\[SWS_DM_01398\]](#)

modes (optional) – hint: the SOVD operation mode is **not yet supported**.

]

7.3.3.5.5 Operations – Start execution of an operation

[SWS_DM_01396] Support of SOVD method Start Execution of an Operation

Upstream requirements: [RS_Diag_04262](#)

[For each [DiagnosticSovdOperationStart](#), the [Diagnostic Server instance](#) shall provide the SOVD method `Start Execution of an Operation` according to ASAM SOVD[5][14].]

[SWS_DM_02174] Operation start execution

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method to start an operation is received and, if needed, the proximity_challenge validation according to [SWS_DM_02173] succeeded, the DM shall call the `ara::diag::GenericDiagnosticSovdOperation::StartExecution`.]

[SWS_DM_02176] Response body of a synchronous operation start

Upstream requirements: [RS_Diag_04262](#)

[If a synchronous SOVD operation has been started, the DM shall return the SOVD response body according to ASAM SOVD[5][14] and set the `parameters` attribute to the value derived from the [DiagnosticSovdContentData](#) within the returned [StartSynchronousExecutionResponse](#) from `ara::diag::GenericDiagnosticSovdOperation::StartExecution`.]

[SWS_DM_02177] Response body of an asynchronous operation start

Upstream requirements: [RS_Diag_04262](#)

[If an asynchronous SOVD operation has been started, the DM shall return the SOVD response body according to ASAM SOVD[5][14] and set the attributes

id to a value provided by the DM to uniquely identify the running operation (concrete value is implementation specific and not defined by AUTOSAR). The **id** is used as `execution-id` for capability calls of this operation.

status to the value derived from [SovdOperationExecution::ExecuteCapability](#) (obtained from returned [StartAsynchronousExecutionResponse](#)).

]

Multiple operations, simultaneously

The ASAM SOVD standard allows multiple instances of the same operation being triggered simultaneously. Once started, the operations are running and can get queried using the **id** attribute in the response body.

It is up to the application to decide, if an operation allows more than a single instance and return an appropriate SOVD error code in case, only a single instance is possible.

It is assumed that operations often only allow a single instance.

7.3.3.5.5.6 Operations – Terminate execution of an operation

[SWS_DM_01392] Support of SOVD method Terminate execution of an Operation

Upstream requirements: [RS_Diag_04262](#)

[For each [DiagnosticSovdOperationStop](#), the [Diagnostic Server instance](#) shall provide the SOVD method `Terminate` execution of an `Operation` according to ASAM SOVD[5][14].]

[SWS_DM_01391] SOVD method Terminate execution of an Operation – proximity_response

Upstream requirements: [RS_Diag_04262](#)

[The `proximity_response` in the request body of SOVD method `Terminate` the `Execution of an Operation` shall be validated using [ara::diag::SovdProximityChallenge::ValidateResponse](#).]

[SWS_DM_02179] Operation terminate execution

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method to terminated an operation is received and, if needed, the `proximity_challenge` validation according to [SWS_DM_02173] succeeded, the DM shall call [ara::diag::SovdOperationExecution::Terminate](#) of the [SovdOperationExecution](#) assigned to the `execution-id` [SWS_DM_02177] of the request body.

Terminating an [SovdOperationExecution](#) will discard the [SovdOperationExecution](#) instance and further method calls on the `execution-id` are no longer possible and the result of such a call is **undefined**.]

7.3.3.5.5.7 Operations – Read the status of an operation execution

[SWS_DM_01393] Support of SOVD method Read the Status of an Operation Execution

Upstream requirements: [RS_Diag_04262](#)

[For each [DiagnosticSovdOperationGetStatus](#), the [Diagnostic Server instance](#) shall provide the SOVD method `Read the Status of an Operation Execution` according to ASAM SOVD[5][14].]

[SWS_DM_02180] Operation Read the Status of an operation

Upstream requirements: [RS_Diag_04262](#)

[If a SOVD method `Read the Status of an operation` is received and, if needed, the `proximity_challenge` validation according to [\[SWS_DM_02173\]](#) succeeded, the DM shall call `SovdOperationExecution::GetStatus` of the `SovdOperationExecution` assigned to the `execution-id` [\[SWS_DM_02177\]](#) of the request body.]

[SWS_DM_02181] Operation status

Upstream requirements: [RS_Diag_04262](#)

[If the DM instance has received the SOVD method `Read the Status of an operation` and the `SovdOperationExecution::GetStatus` method has succeeded, the DM shall send a response body according to ASAM SOVD[5][14] with the attributes:

status set to value, derived from `SovdOperationExecution::GetStatus` (obtained from returned `GetStatusResponse.status`),

capbaility set to `execute`,

parameters set to values, derived from the `DiagnosticSovdContentData` (via `GetStatusResponse.Parameters`).

]

7.3.3.5.6 SOVD API – Modes**Supported Modes by native SOVD**

`CommunicationControl` and `ControlDTCSetting` are the only supported modes relevant to native SOVD.

[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[5].]

7.3.3.5.6.1 Modes – 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[5].]

[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.]

7.3.3.5.6.2 Modes – 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[5].]

[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].]

7.3.3.5.7 SOVD API – 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, that have been introduced by UDS Service 0x38 – RequestFileTransfer and are described in detail in section [Chapter 7.3.2.8.20](#).

[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 [5].]

[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 [\[5\]](#).]

[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::GetBulkDataMeta-Data.](#)]

[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::DeleteAll-BulkData.](#)]

[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.](#)]

7.3.3.5.8 SOVD API – Software Update

This section describes realization of the SOVD API Methods for Software Update. While the ASAM SOVD[\[5\]](#) 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.

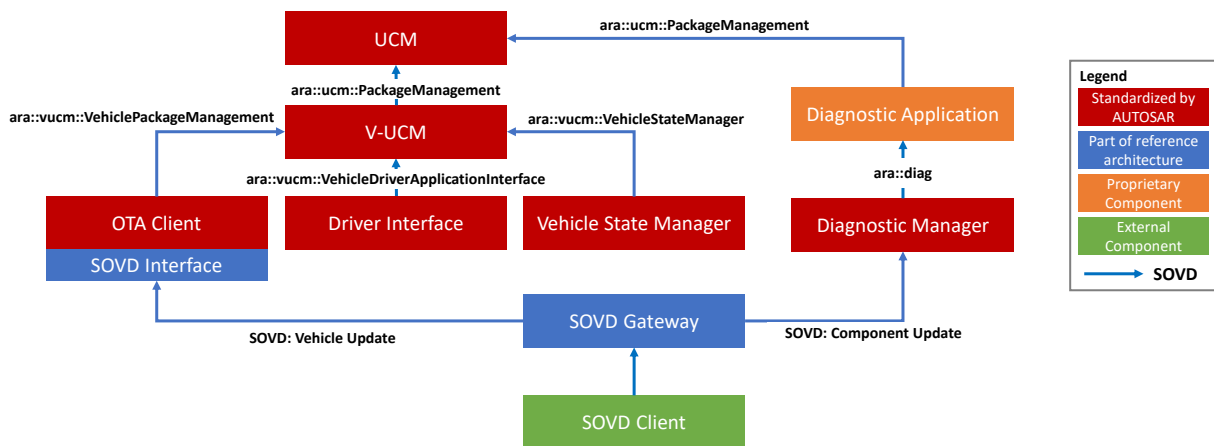


Figure 7.4: SOVD Software Update API - concept. Figure elaborates on the difference of vehicle level and component level SOVD Software Update API.

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 7.3.2.8.20.

[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 [5].]

[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.

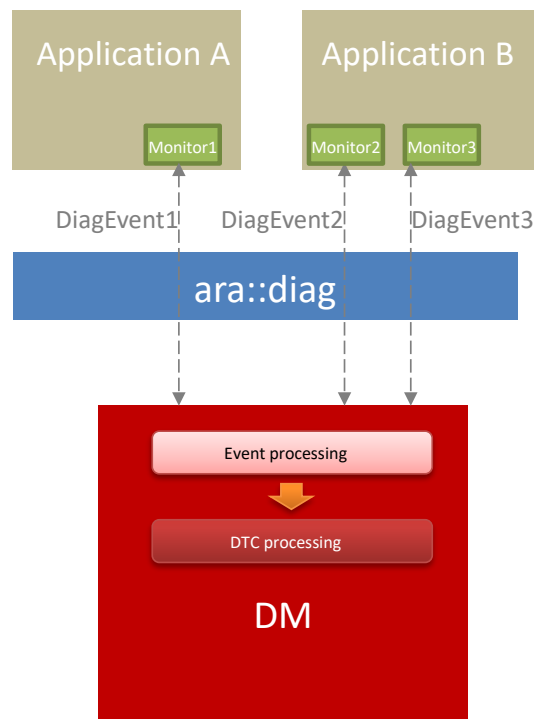


Figure 7.5: Example of diagnostic event usage

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/de-initialization 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 course 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 an `operation cycle` is restarted by calling the method `ara::diag::OperationCycle::RestartOperationCycle`, for all monitors that are mapped via `DiagnosticEventToOperationCycleMapping` to the restarted `operation cycle`, the DM shall call the registered notifier method `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

Upstream requirements: [RS_Diag_04179](#)

[

...	Debouncing method		
	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

Upstream requirements: [RS_AP_00142](#)

[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

Upstream requirements: [RS_Diag_04200](#)

[

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

Upstream requirements: [RS_Diag_04200](#)

[

UDS status bit description		Combined UDS status information logical equation				
0	TestFailed	$\text{CbDTC}_{\text{Bit0}} = \text{Event}[1]_{\text{Bit0}} \mid \text{Event}[2]_{\text{Bit0}} \mid \dots \mid \text{Event}[n]_{\text{Bit0}}$				
1	TestFailedThis OperationCycle	$\text{CbDTC}_{\text{Bit1}} = \text{Event}[1]_{\text{Bit1}} \mid \text{Event}[2]_{\text{Bit1}} \mid \dots \mid \text{Event}[n]_{\text{Bit1}}$				
2	PendingDTC	$\text{CbDTC}_{\text{Bit2}} = \text{Event}[1]_{\text{Bit2}} \mid \text{Event}[2]_{\text{Bit2}} \mid \dots \mid \text{Event}[n]_{\text{Bit2}}$				
3	ConfirmedDTC	$\text{CbDTC}_{\text{Bit3}} = \text{Event}[1]_{\text{Bit3}} \mid \text{Event}[2]_{\text{Bit3}} \mid \dots \mid \text{Event}[n]_{\text{Bit3}}$				
4	TestNotCompleted SinceLastClear	$\text{CbDTC}_{\text{Bit4}} = (\text{Event}[1]_{\text{Bit4}} \mid \text{Event}[2]_{\text{Bit4}} \mid \dots \mid \text{Event}[n]_{\text{Bit4}}) \& ! \text{CbDTC}_{\text{Bit5}}$				
5	TestFailedSince LastClear	$\text{CbDTC}_{\text{Bit5}} = \text{Event}[1]_{\text{Bit5}} \mid \text{Event}[2]_{\text{Bit5}} \mid \dots \mid \text{Event}[n]_{\text{Bit5}}$				
6	TestNotCompleted ThisOperationCycle	$\text{CbDTC}_{\text{Bit6}} = (\text{Event}[1]_{\text{Bit6}} \mid \text{Event}[2]_{\text{Bit6}} \mid \dots \mid \text{Event}[n]_{\text{Bit6}}) \& ! \text{CbDTC}_{\text{Bit1}}$				
7	WarningIndicator Requested	$\text{CbDTC}_{\text{Bit7}} = \text{Event}[1]_{\text{Bit7}} \mid \text{Event}[2]_{\text{Bit7}} \mid \dots \mid \text{Event}[n]_{\text{Bit7}}$				

]

A meaningful use of event combination requires that certain configuration values of individual events or related properties have the same configured values. In particular the diagnostic indicator configuration for combined events requires this. These consistent settings is enforced by constraints in the DEXT.

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 `reportingInChronologicalOrderOldestFirst` 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`. Their 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 condition states. 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 condition-Handle 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 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 [DiagnosticEvent-ToDebounceAlgorithmMapping](#), 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.]

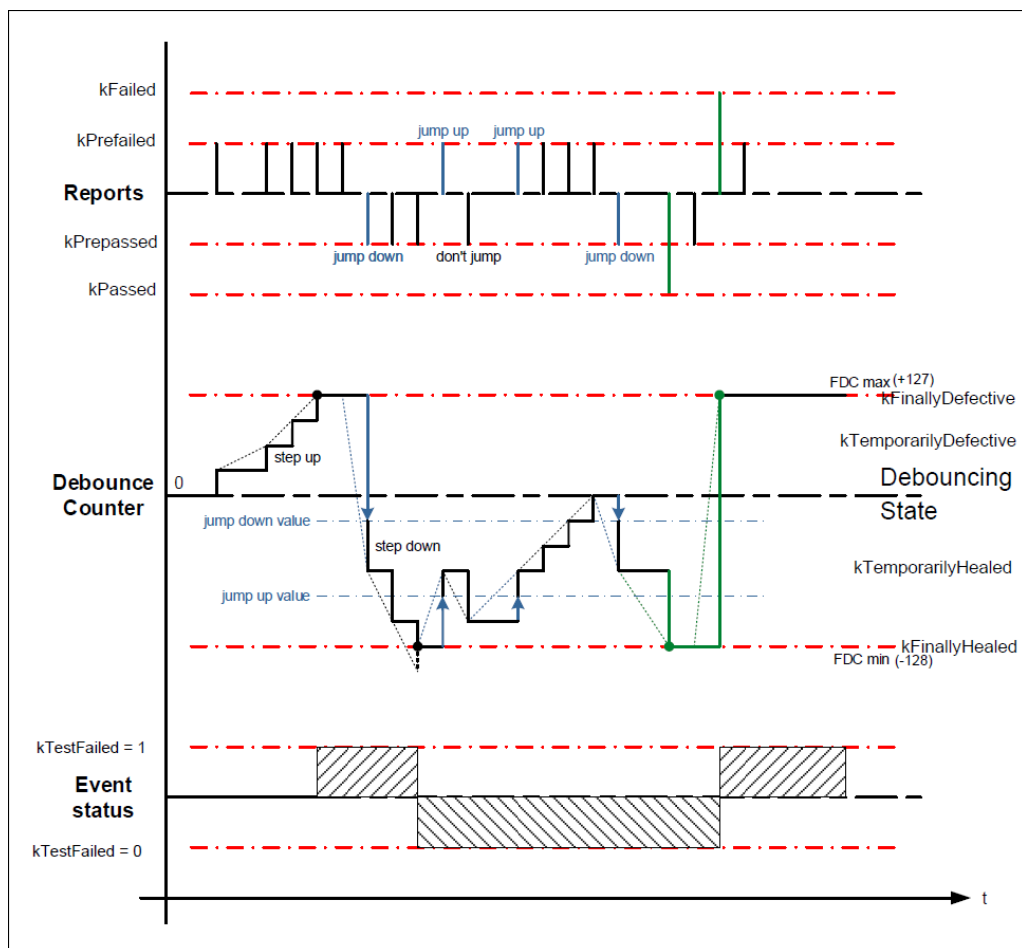


Figure 7.6: Counter-based debouncing

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`.]

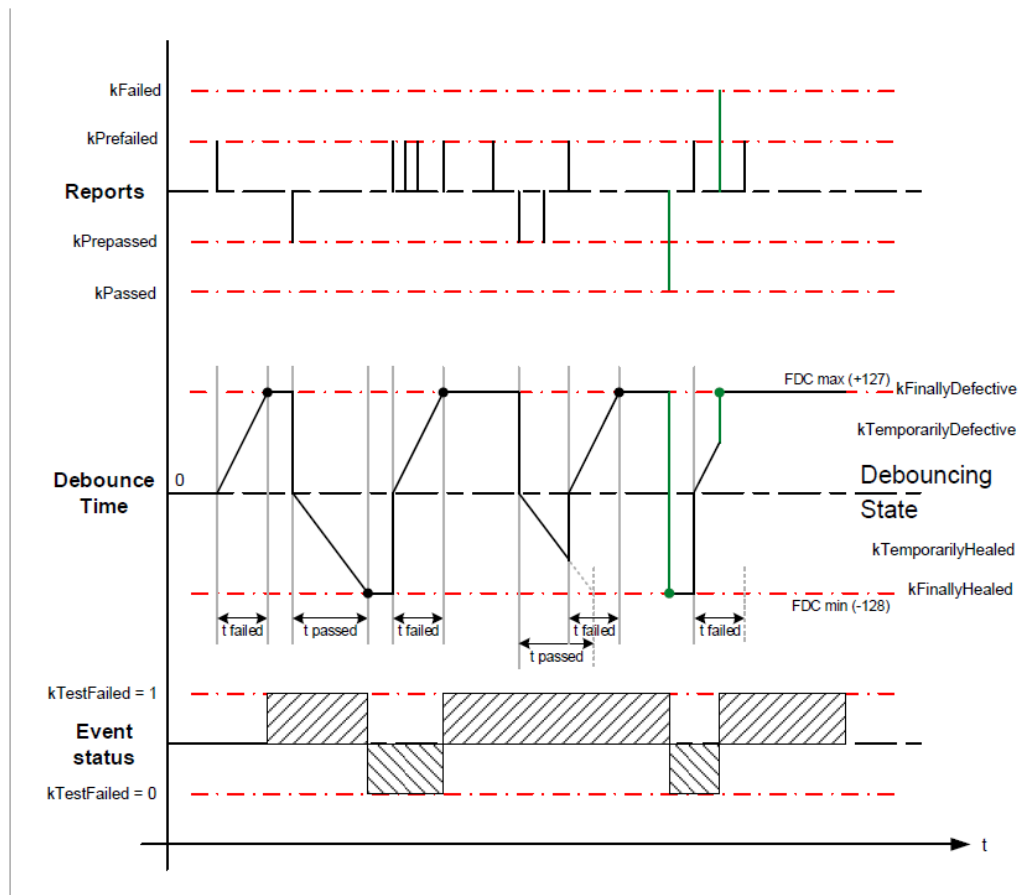


Figure 7.7: Timer based debouncing

[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 `kFreezeDebounce`, 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 `kFreezeDebounce`) 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

Upstream requirements: [RS_Diag_04068](#)

[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.diagnosticOperationCycle](#).]

[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 [4].

[SWS_DM_CONSTR_00059] Restriction on supported DTC format [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 0xFFFFF.

[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 [4].

[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](#) 0xFFFFFFFF. This DTC group contains always all configured DTCs.]

[SWS_DM_CONSTR_00082] Restriction on the configuration of the DTC group GroupOfAllDTCs [The DM shall ignore any configuration of a [DiagnosticTroubleCodeGroup.groupNumber](#) with a value of 0xFFFFFFFF.]

A configuration of the DTC group 0xFFFFFFFF 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 [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 triggers 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 corresponding 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](#)

]

[SWS_DM_01582] Healing counter increment

Upstream requirements: [RS_Diag_04204](#)

[If the [DiagnosticOperationCycle](#) of the [DiagnosticEvent](#) mapped by [DiagnosticEventToOperationCycleMapping](#) is restarted and the [event status bit](#) [TestFailedThisOperationCycle](#) and [TestNotCompletedThisOperationCycle](#) both are set to 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](#)

AA controlled WIR status bit handling is not foreseen.

Setting the WIR-bit of a DTC is controlled by the DM internal WIR-bit handling.

AA can retrieve the WIR status bit, using the [ara::diag::Event::GetEventStatus](#) function.

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 [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.]

7.3.4.4.1 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 [\[4\]](#)).]

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.]

7.3.4.4.2 Storage of [snapshot record](#) data

The fault memory manager provided by the DM is designed to be diagnostic protocol independent. It receives states from monitors, operation cycles or conditions. Based on this information it calculates DTC states, does aging, etc. Upon requests from a diagnostic protocol the fault memory manager can provide the corresponding information based on all the stored information. However for snapshot records (and as well for extended data records) there is a protocol dependency due to the fact, that UDS snapshot records contain DataIdentifiers (DIDs) and SOVD snapshot records contain SOVD data. While both are of similar nature and provided as [environmental data](#) at a time a fault occurred, the interfaces towards the application to capture this data have a protocol specific design (e.g. having UDS NRC code as possible return values of a Read operation).

As result the DM uses two different snapshot records definitions for a DTC. One used only for UDS which is defined by the [DiagnosticTroubleCodeProps.snapshotRecordContent](#) and another used for SOVD which is defined by [DiagnosticTroubleCodeProps.sovdSnapshotContent](#). While complex SOVD data in general cannot be mapped to UDS in all situations (e.g. stored video stream), the DM does only consider the UDS based [DiagnosticTroubleCodeProps.snapshotRecordContent](#) for UDS snapshot reporting. For SOVD both ways are supported. A SOVD snapshot record will report the data stored for [DiagnosticTroubleCodeProps.snapshotRecordContent](#) and [DiagnosticTroubleCodeProps.sovdSnapshotContent](#). In that case the UDS snapshot data is transformed to SOVD by using the shortname of the [DiagnosticDataElement](#) as name and the value is the Read 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 records 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 records 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] UDS Snapshot record layout

Upstream requirements: [RS_Diag_04205](#)

[If any [snapshot record](#) storage trigger occurs, the DM shall capture the UDS snapshot record data defined by [DiagnosticTroubleCodeProps.snapshotRecordContent](#). Each referenced [DiagnosticDataIdentifier](#) shall be captured via the [PortPrototype](#) configured for these DIDs.]

[SWS_DM_02142] SOVD Snapshot record layout

Upstream requirements: [RS_Diag_04205](#)

[If any [snapshot record](#) storage trigger occurs, the DM shall capture the SOVD snapshot record data defined by [DiagnosticTroubleCodeProps.sovdSnapshotContent](#). Each referenced [DiagnosticSovdContent](#) shall be captured via the [PortPrototype](#) configured for these SOVD content group.]

All configured UDS and SOVD snapshot record data are captured if the storage trigger occurs.

[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.]

7.3.4.4.3 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](#) [DiagnosticExtendedDataRecord.element](#) 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 [DiagnosticExtendedDataRecordElement](#) referenced from [DiagnosticEdrDataProviderMapping](#), 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 UDS to other [DiagnosticDataElement](#) data such as from [snapshot records](#) or [DiagnosticDataElement](#) from DIDs or SOVD data.

[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](#).]

The nature of data stored and retrieved via extended data records is typical of standard scalar data types. Mostly extended data records represent counters or comparable statistic data related to a DTC. While a snapshot record could contain complexer data information at the time a fault was detected, the extended data record is simpler. This is reflected also within the interface.

The typed and generated `extendend-data-record-element-interface-name-x` class has an interface that is designed to meet the purpose providing these simple data information from an application.

The main difference to UDS `DiagnosticDataElement` read operation is, that extended data record data shall always be available and no `NRC` as return value is supported. There are no known use cases for applications returning extended data to not provide data to the read request. Dedicated invalid value information, such as the use of `0xFF` for *invalid* can be used, if this is needed. Especially the UDS protocol has no means to omit data in the positive response stream. A UDS client relies, that data positions are kept in the response, especially if multiple data elements are part of the response. Here data elements with no value cannot be serialized to the UDS response message.

[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.element`. Each `DiagnosticExtendedDataRecordElement` shall be captured in its order via the corresponding read function instance for typed `ExtendedDataRecordElement`: `<namespace-list-extended-data-record-element>::<extendend-data-record-element-interface-name-x>::Read.`]

7.3.4.4.4 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` ([7.3.2.8.7.5](#)).

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 [DiagnosticExtendedDataRecordElement](#) is referenced by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) is set to `DEM_DTC_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 [DiagnosticExtendedDataRecordElement](#) is referred by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) is set to `DEM_OVFLIND`, the DM shall set the value of this [internal data element](#) to the 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 [DiagnosticExtendedDataRecordElement](#) is referred by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) 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 [DiagnosticExtendedDataRecordElement](#) is referred by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) 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 [DiagnosticExtendedDataRecordElement](#) is referred by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) 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 [DiagnosticExtendedDataRecordElement](#) is referred by [DiagnosticEdrDataProviderMapping.extendedDataRecordElement](#) and [DiagnosticEdrDataProviderMapping.dataProvider](#) 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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticE-drDataProviderMapping.extendedDataRecordElement` and `DiagnosticE-drDataProviderMapping.dataProvider` 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 [\[SWS_DM_00957\]](#).

The length of this `internal data element` is one byte.]

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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticE-drDataProviderMapping.extendedDataRecordElement` and `DiagnosticE-drDataProviderMapping.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.

The length of this `internal data element` is one byte.]

[SWS_DM_00957] Configuration of DTC "aging counter down" as extended data record

Upstream requirements: [RS_Diag_04190](#)

[If the `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticE-drDataProviderMapping.extendedDataRecordElement` and `DiagnosticE-drDataProviderMapping.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.

The length of this `internal data element` is one byte.]

[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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` is set to `DEM_AGINGCTR_UPCNT`, the DM shall set the respective extended data record 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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` is set to `DEM_AGINGCTR_DOWNCNT`, the DM shall set the respective extended data record value to `DiagnosticAging.threshold` if configured or '255' otherwise.]

[SWS_DM_CONSTR_00960] No support for `DEM_AGINGCTR_UPCNT_FIRST_ACTIVE`

[`DiagnosticEdrDataProviderMapping.dataProvider` set to `DEM_AGINGCTR_UPCNT_FIRST_ACTIVE` referencing a `DiagnosticExtendedDataRecordElement` 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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 extended data record.

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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 extended data record.

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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 extended data record.

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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 extended data record.

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 `DiagnosticExtendedDataRecordElement` is referred by `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` and `DiagnosticEdrDataProviderMapping.dataProvider` 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 extended data record.

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] Availability 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 (`kConditionsNotCorrect`).]

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 [7.3.2.8.6.1](#) 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 `DiagnosticEdrDataProviderMapping.dataProvider` of the corresponding `DiagnosticEdrDataProviderMapping.extendedDataRecordElement` 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 occurrence

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.

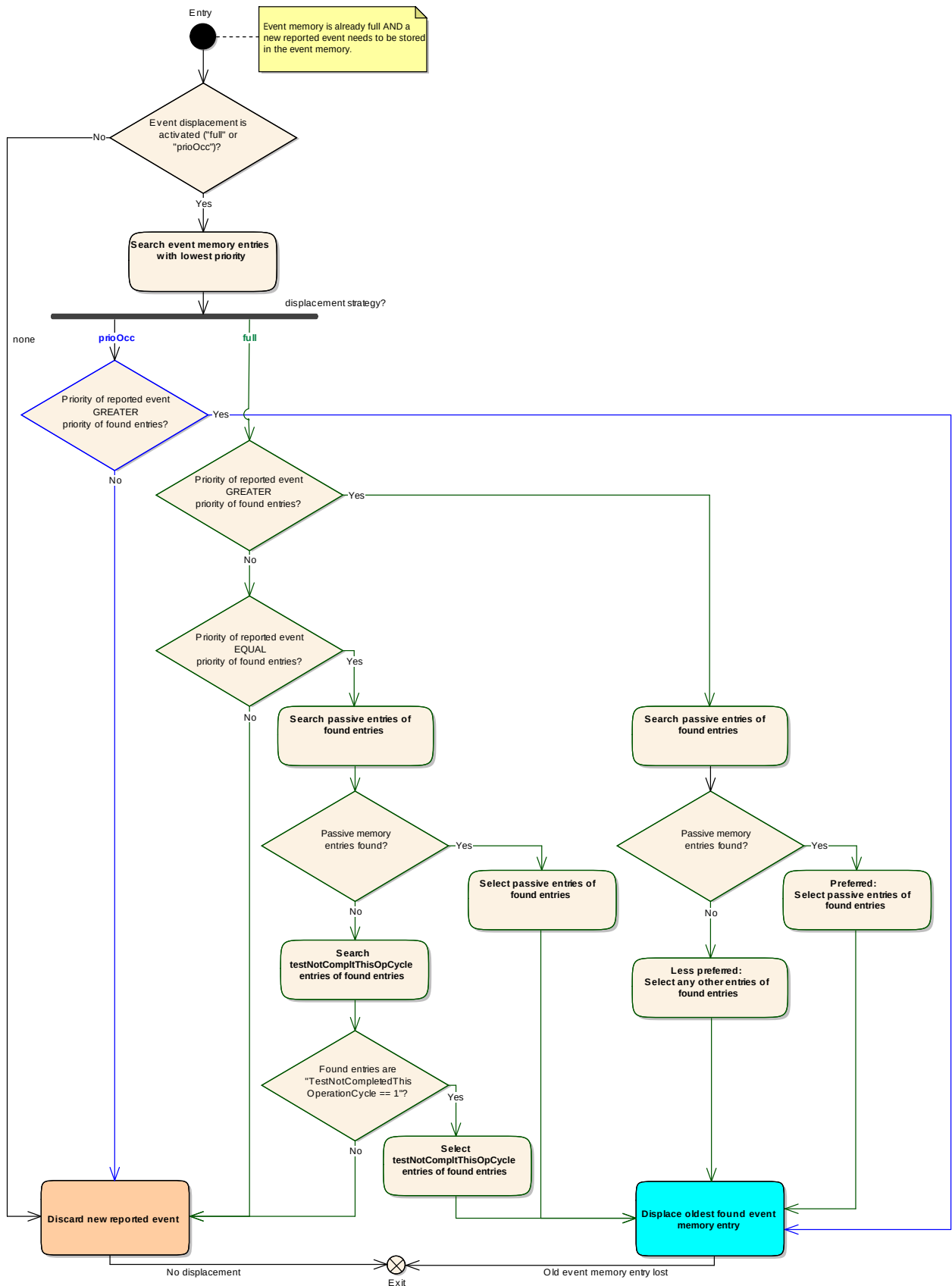


Figure 7.8: Combined displacement criteria processing

[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 chronological 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) [\[4\]](#) 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 [\[4\]](#) 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 [\[4\]](#) is used to configure the DM, a certain minimum content is required.

In this chapter a loose list of DEXT [\[4\]](#) 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
[Each `DiagnosticEvent` requires exactly one `DiagnosticEventToOperationCycleMapping`, referencing the `diagnosticEvent` and one `DiagnosticOperationCycle`.]

[SWS_DM_CONSTR_00206] Supported format for data identifier for VINDataIdentifier
[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

There are two types of data elements in the context of the DM:

An external data element is used to read data from an external application. Internal data elements have data providers from within the DM. In these cases the data is already known and maintained by the DM and there is no reason to delegate the read request to an application.

Internal data elements in the context of the fault memory are used within extended data record elements (`DiagnosticExtendedDataRecordElement`) that get their data source via `DiagnosticEdrDataProviderMapping.dataProvider` and referring to a `DiagnosticExtendedDataRecordElement`. Internal data elements in context of the diagnostic UDS communication can

be used by [DiagnosticProvidedDataMapping](#), referencing this [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](#).

[SWS_DM_01565] Supported [internal](#) data for UDS [ReadDataByIdentifier](#)

Upstream requirements: [RS_Diag_04097](#)

[

Data Provider	Content	Format
DCM_SESSION	Current session of contextual Diagnostic Conversation	1 byte
DCM_SECURITY_LEVEL	Current security level of contextual Diagnostic Conversation	1 byte

]

[SWS_DM_02145] gives a list of the supported [internal DiagnosticExtendedDataRecordElements](#), where

Data Provider refers to the NameToken defined by the attribute [dataProvider](#) of the associated [DiagnosticEdrDataProviderMapping](#),

Content describes the actual content of the data,

Format describes the data format of the [DiagnosticExtendedDataRecordElement](#).

[SWS_DM_02145] Support of **internal** data for ExtendDataRecords

Upstream requirements: [RS_Diag_04097](#)

[

Data Provider	Content	Format
DEM_AGINGCTR_DOWNCNT	Down-counting aging counter of contextual DTC.	1 byte
DEM_AGINGCTR_UPCNT	Up-counting aging counter of contextual DTC.	1 byte
DEM_CURRENT_FDC	Fault Detection Counter of contextual DTC.	1 byte
DEM_CYCLES_SINCE_FIRST_FAILED	Operation Cycle Counter of contextual DTC – Operation Cycles Since First Failed.	1 byte
DEM_CYCLES_SINCE_LAST_FAILED	Operation Cycle Counter of contextual DTC – Operation Cycles Since Last Failed.	1 byte
DEM_FAILED_CYCLES	Operation Cycle Counter of contextual DTC – Failed Operation Cycles.	1 byte
DEM_MAX_FDC_DURING_CURRENT_CYCLE	Fault Detection Counter maximum value during current operation cycle of contextual DTC.	1 byte
DEM_MAX_FDC_SINCE_LAST_CLEAR	Fault Detection Counter maximum value since last clear of contextual DTC.	1 byte
DEM_OCCCTR	Occurrence Counter of contextual DTC.	1 byte
DEM_OVFLIND	Overflow indication of contextual DTCs error memory: (0 = FALSE, 1 = TRUE)	1 byte
DEM_SIGNIFICANCE	Event significance of contextual DTC (refer to DemDTCSignificance) (0 = OCCURRENCE, 1 = FAULT)	1 byte
DEM_PRIORITY	Priority of the contextual DTC.	1 byte
DEM_EVENT_ASSOCIATED_IDENTIFICATION	Represents the static value associated to it by associatedEventIdentification .	4 bytes
DEM_CYCLES_TESTED_SINCE_LAST_FAILED	Tested cycles since last failed excluding test not complete. The counter latches at 255.	1 byte
DEM_CYCLES_TESTED_SINCE_FIRST_FAILED	Tested cycles since first failed excluding test not complete. The counter latches at 255.	1 byte

]

[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

[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 response / DEM context.

[SWS_DM_CONSTR_00395] Restriction on DEM-exclusive `DiagnosticDataElements`

[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`

[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 `namespacelist-dataelement::dataelementinterfacenamex` class and call its `namespacelistdataelement::dataelementinterfacenamex::Read` function.]

Note: In general, there are multiple instances of `namespace::dataelement::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 [23].

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 `<namespace-list-data-identifier>::<data-identifier-interface-name-x>` 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 `<namespace-list-data-identifier>::<data-identifier-interface-name-x>` 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_Diag_04211](#), [RS_Diag_04246](#)

[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 [Upon reception of SIGTERM the [Diagnostic Server instance](#) shall gracefully shutdown itself.]

[SWS_DM_01573] Stop all running [Transport Protocol Handlers](#)

Upstream requirements: [RS_Diag_04147](#), [RS_Diag_04168](#)

[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_Diag_04211](#), [RS_Diag_04246](#)

[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
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	Successfully 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 ClearDiagnosticInformation.	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
SEV_DOIP_HEADER_CHECK_FAILED	The DoIP Header Handler rejected a request (routing or diagnostic message).	127
SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED	A routing request was rejected by the routing handler.	128





Name	Description	ID
SEV_DOIP_ROUTING_ACTIVATION_SUCCESS	A routing request was successful.	129
SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED	A diagnostic message request was rejected by the diagnostic message handler.	130
SEV_ACCESS_CONTROL_DM_IAM_ACCESS_DENIED	Access of an application to a resource provided by DM was denied.	133

]

In situations when a single diagnostic request can result in more than one security event, the Diagnostic Server Instance can report more than one security event.

Example: The diagnostic service 0x2E (WriteDataByIdentifier) as rejected with NRC 0x33 (SecurityAccessDenied) and the DM is configured to report SEV_UDS_SECURITY_ACCESS_NEEDED and SEV_UDS_WRITE_DATA_FAILED. In this case the two security events SEV_UDS_SECURITY_ACCESS_NEEDED and SEV_UDS_WRITE_DATA_FAILED are reported.

[SWS_DM_02015] Reporting security access denied to IdsM

Upstream requirements: [RS_Ids_00810](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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	
NegativeResponseCode	uint8	

]

[SWS_DM_02023] Reporting successful authentication to IdsM*Upstream requirements:* [RS_Ids_00810](#)

[If 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 [26] 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	Successfully 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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	
Context Data	Data Type	Allowed Values





SEV Name	SEV_UDS_REQUEST_FILE_TRANSFER_SUCCESSFUL	
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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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
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](#)

[If 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 [26] 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

[If 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 [26] 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

[If 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 [26] 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](#)

[If 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 [26] 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](#)

[If 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 [26] 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
RID	uint16	
Subfunction	uint8	
ClientSourceAddress	uint16	

]

[SWS_DM_02057] Reporting routine control failed to IdsM

Upstream requirements: RS_Ids_00810

[If 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 [26] 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	

]

[SWS_DM_02132] Reporting DM access control denied to IdsM

Upstream requirements: [RS_Ids_00810](#)

[If a process interacts with ara:diag, and access restrictions apply, the DM shall report a security event SEV_ACCESS_CONTROL_DM_IAM_DENIED to IdsM (see [26] and table [SWS_DM_02133]) with the context data given in [SWS_DM_02133].]

[SWS_DM_02133] Security event context data definition: SEV_ACCESS_CONTROL_DM_IAM_ACCESS_DENIED

Status: DRAFT

Upstream requirements: [RS_Ids_00810](#)

[

SEV Name	SEV_ACCESS_CONTROL_DM_IAM_ACCESS_DENIED	
ID	133	
Description	Access of an application to a resource provided by DM was denied.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
UserId	uint32	

]

[SWS_DM_02134] Reporting "DoIP header check failed" to IdsM

Upstream requirements: [RS_Ids_00810](#)

[If a DoIP message is rejected based on an invalid "DoIP generic header", the DM shall report a security event SEV_DOIP_HEADER_CHECK_FAILED to IdsM (see [26] and table [SWS_DM_02135]) with the context data given in [SWS_DM_02135].]

The DoIP generic header check is described in ISO 13400-2 in the figure "DoIP generic header handler".

[SWS_DM_02135] Security event context data definition: SEV_DOIP_HEADER_CHECK_FAILED

Status: DRAFT

Upstream requirements: [RS_Ids_00810](#)

[

SEV Name	SEV_DOIP_HEADER_CHECK_FAILED	
ID	127	
Description	The DoIP Header Handler rejected a request (routing or diagnostic message).	
Context Data Version	1	
Context Data	Data Type	Allowed Values
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
ProtocolVersion	uint8 [1]	
PayloadType	uint8 [2]	
NACKCode	uint8 [1]	

]

[SWS_DM_02136] Reporting "DoIP routing request rejected" to IdsM*Upstream requirements:* [RS_Ids_00810](#)

[If a "DoIP routing activation request" does not lead to a "routing activation response code" that activates routing, the DM shall report a security event SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED to IdsM (see [26] and table [SWS_DM_02137]) with the context data given in [SWS_DM_02137].]

The DoIP routing activation check is described in ISO 13400-2 in the figure "DoIP routing activation handler".

[SWS_DM_02137] Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED*Status:* DRAFT*Upstream requirements:* [RS_Ids_00810](#)

SEV Name	SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED	
ID	128	
Description	A routing request was rejected by the routing handler.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
ActivationType	uint8 [1]	
ResponseCode	uint8 [1]	

[SWS_DM_02138] Reporting "DoIP routing request accepted" to IdsM*Upstream requirements:* [RS_Ids_00810](#)

[If a "DoIP routing activation request" leads to a "routing activation response code" that activates routing, the DM shall report a security event SEV_DOIP_ROUTING_ACTIVATION_SUCCESS to IdsM (see [26] and table [SWS_DM_02139]) with the context data given in [SWS_DM_02139].]

[SWS_DM_02139] Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_SUCCESS*Status:* DRAFT*Upstream requirements:* [RS_Ids_00810](#)

SEV Name	SEV_DOIP_ROUTING_ACTIVATION_SUCCESS	
ID	129	
Description	A routing request was successful.	
Context Data Version	1	





SEV Name	SEV_DOIP_ROUTING_ACTIVATION_SUCCESS	
Context Data	Data Type	Allowed Values
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
ActivationType	uint8 [1]	

]

[SWS_DM_02140] Reporting "DoIP diagnostic message check failed" to IdsM

Upstream requirements: [RS_Ids_00810](#)

[If a DoIP diagnostic message is rejected and leads to a "diagnostic message negative acknowledgment", the DM shall report a security event SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED to IdsM (see [26] and table [SWS_DM_02141]) with the context data given in [SWS_DM_02141].]

The DoIP diagnostic message check is described in ISO 13400-2 in the figure "DoIP diagnostic message handler".

[SWS_DM_02141] Security event context data definition: SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED

Status: DRAFT

Upstream requirements: [RS_Ids_00810](#)

[

SEV Name	SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED	
ID	130	
Description	A diagnostic message request was rejected by the diagnostic message handler.	
Context Data Version	1	
Context Data	Data Type	Allowed Values
SourceIp	uint8 [16]	
SourcePort	uint8 [2]	
LogicalSourceAddress	uint8 [2]	
LogicalTargetAddress	uint8 [2]	
NACKCode	uint8 [1]	

]

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.

Please note that concrete implementations might additionally implement Non-Standardized Violations (see also [SWS_CORE_00003]).

[SWS_CORE_13003]

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.		
MessageId	0x80001ffc		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	ArgumentDescription	ArgumentType	ArgumentUnit
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13004]

Dlt-Message	PortInterfaceMappingViolation		
Description	The type of mapping does not match the expected type of PortInterface: {portInterfaceTypeName} referenced by a {mappingTypeName}.		
MessageId	0x80001ffb		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	ArgumentDescription	ArgumentType	ArgumentUnit
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13005]

Dlt-Message	ProcessMappingViolation		
Description	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.		
MessageId	0x80001ffa		





MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	ArgumentDescription	ArgumentType	ArgumentUnit
modeledProcess Id	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13006]

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.		
MessageId	0x80001ff9		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	ArgumentDescription	ArgumentType	ArgumentUnit
modeledProcess Id	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[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.		
MessageId	0x80003fff		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	ArgumentDescription	ArgumentType	ArgumentUnit
modeledProcess Id	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	





modelled Algorithm	Name of the mapped debounce algorithm according to DiagnosticEventToDebounceAlgorithmMapping.	uint8 [encoding UTF-8]	
provided Algorithm	Name of the algorithm provided during object construction.	uint8 [encoding UTF-8]	

]

[SWS_DM_02062] ViolationMessage UnexpectedMonitorActionHandlingViolation

Status: DRAFT

Upstream requirements: [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
modeledProcess Id	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
monitorAction	Value of the requested "action" parameter of ara::diag::Monitor::ReportMonitorAction.	uint8 [encoding UTF-8]	
debouncing Method	Value of the used debouncing method.	uint8 [encoding UTF-8]	

]

7.5.4 Production Errors

This functional cluster does not define any production errors (i.e., Diagnostic Events).

7.6 Access Control

The generation of security events, modification of context data, and interaction over any ara::diag interface is subject to access control, i.e., it can be restricted which processes can perform these tasks.

[SWS_DM_02130] Access control on ara::diag interfaces

Upstream requirements: [RS_Ids_00810](#)

[The [Diagnostic Server instance](#) shall restrict the processes that can interact over any ara::diag interface only to the processes referenced by the corresponding mapping in the role process in the manifest. Restriction means that no interaction

with the initiating process shall be done and - if configured - a security event shall be raised.]

[SWS_DM_02131] Security event for access control

Upstream requirements: [RS_Ids_00810](#)

[The DM shall raise SEV_ACCESS_CONTROL_DM_IAM_ACCESS_DENIED security event in case of any violations mentioned in [\[SWS_DM_02130\]](#).]

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)	
Port Interfaces:	States that the C++ API <code>class</code> is the related C++ API binding for the given modeled sub-class of <code>PortInterface</code>	
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 C++ <code>namespace</code> (in case of a <code>class</code> or non-member function) or a <code>class</code> declaration (in case of a member)	
Symbol:	C++ symbol 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 parameterize 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 C++ <code>Exceptions</code> that may be thrown by the function	
Violations:	List of violations that may raised by the function	
Errors:	Error type (0..*)	List of defined <code>ara::core::ErrorCodes</code> 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 standardized by AUTOSAR
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

8.1 PortInterface to API class binding

This table shows the `API class` binding for each `PortInterface` owned by this functional cluster and those functions taking an `ara::core::InstanceSpecifier` argument, designated to "construct" that class. These constructing functions may be any combination of special-member constructors, named constructor members or non-member factory constructors.

Port Interface	API Class / Function
DiagnosticAuthenticationInterface	[SWS_DM_01123] Definition of API class ara::diag::Authentication
	[SWS_DM_01124] Definition of API function ara::diag::Authentication::Authentication
DiagnosticComControlInterface	[SWS_DM_00804] Definition of API class ara::diag::CommunicationControl
	[SWS_DM_00806] Definition of API function ara::diag::CommunicationControl::CommunicationControl
DiagnosticConditionInterface	[SWS_DM_00711] Definition of API class ara::diag::Condition
	[SWS_DM_00712] Definition of API function ara::diag::Condition::Condition
DiagnosticDTCInformationInterface	[SWS_DM_00657] Definition of API class ara::diag::DTCInformation
	[SWS_DM_00664] Definition of API function ara::diag::DTCInformation::DTCInformation
DiagnosticDataElementInterface	[SWS_DM_00603] Definition of API class {<namespace-list-data-element>>::{<data-element-interface-name-x>}
	[SWS_DM_00587] Definition of API function {<namespace-list-data-element>>::{<data-element-interface-name-x>>::{<data-element-interface-name-1>}
DiagnosticDataIdentifierGenericInterface	[SWS_DM_00607] Definition of API class ara::diag::GenericDataIdentifier
	[SWS_DM_00634] Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier
DiagnosticDataIdentifierInterface	[SWS_DM_00601] Definition of API class {<namespace-list-data-identifier>>::{<data-identifier-interface-name-x>}
	[SWS_DM_00585] Definition of API function {<namespace-list-data-identifier>>::{<data-identifier-interface-name-x>>::{<data-identifier-interface-name-1>}
DiagnosticDoIPActivationLineInterface	[SWS_DM_00830] Definition of API class ara::diag::DoIPActivationLine
	[SWS_DM_00831] Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
DiagnosticDoIPEntityIdentificationInterface	[SWS_DM_01362] Definition of API class ara::diag::DoIPEntityIdentification
	[SWS_DM_01364] Definition of API function ara::diag::DoIPEntityIdentification::DoIPEntityIdentification
DiagnosticDoIPGroupIdentificationInterface	[SWS_DM_00720] Definition of API class ara::diag::DoIPGroupIdentification
	[SWS_DM_00722] Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification
DiagnosticDoIPPowerModeInterface	[SWS_DM_00731] Definition of API class ara::diag::DoIPPowerMode
	[SWS_DM_00732] Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
DiagnosticDownloadInterface	[SWS_DM_00784] Definition of API class ara::diag::DownloadService
	[SWS_DM_00787] Definition of API function ara::diag::DownloadService::DownloadService
DiagnosticEcuResetInterface	[SWS_DM_01009] Definition of API class ara::diag::EcuResetRequest
	[SWS_DM_01010] Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
DiagnosticEventInterface	[SWS_DM_00646] Definition of API class ara::diag::Event
	[SWS_DM_00647] Definition of API function ara::diag::Event::Event
DiagnosticExternalAuthenticationInterface	[SWS_DM_01191] Definition of API class ara::diag::ExternalAuthentication





Port Interface	API Class / Function
	[SWS_DM_01193] Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication
DiagnosticGenericUdsInterface	[SWS_DM_00602] Definition of API class ara::diag::GenericUDSService [SWS_DM_00616] Definition of API function ara::diag::GenericUDSService::GenericUDSService
DiagnosticIndicatorInterface	[SWS_DM_00741] Definition of API class ara::diag::Indicator [SWS_DM_00742] Definition of API function ara::diag::Indicator::Indicator
DiagnosticMonitorInterface	[SWS_DM_00542] Definition of API class ara::diag::Monitor [SWS_DM_00548] Definition of API function ara::diag::Monitor::Monitor [SWS_DM_01972] 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
DiagnosticMultipleConditionInterface	[SWS_DM_01730] Definition of API class ara::diag::MultipleCondition [SWS_DM_01727] Definition of API function ara::diag::MultipleCondition::MultipleCondition
DiagnosticMultipleEventInterface	[SWS_DM_01704] Definition of API class ara::diag::MultipleEvent [SWS_DM_01705] Definition of API function ara::diag::MultipleEvent::MultipleEvent
DiagnosticMultipleMonitorInterface	[SWS_DM_01694] Definition of API class ara::diag::MultipleMonitor [SWS_DM_01695] Definition of API function ara::diag::MultipleMonitor::MultipleMonitor
DiagnosticOperationCycleInterface	[SWS_DM_00751] Definition of API class ara::diag::OperationCycle [SWS_DM_00752] Definition of API function ara::diag::OperationCycle::OperationCycle
DiagnosticRequestFileTransferInterface	[SWS_DM_01320] Definition of API class ara::diag::FileTransferService [SWS_DM_01325] Definition of API function ara::diag::FileTransferService::FileTransferService
DiagnosticRoutineGenericInterface	[SWS_DM_00605] Definition of API class ara::diag::GenericRoutine [SWS_DM_00552] Definition of API function ara::diag::GenericRoutine::GenericRoutine
DiagnosticRoutineInterface	[SWS_DM_00604] Definition of API class {<routine-interface-hierarchical-namespace-list>::{<routine-interface-name-x>} [SWS_DM_00589] Definition of API function {<routine-interface-hierarchical-namespace-list>::{<routine-interface-name-x>::{<routine-interface-name-1>}}
DiagnosticSecurityLevelInterface	[SWS_DM_00761] Definition of API class ara::diag::SecurityAccess [SWS_DM_00762] Definition of API function ara::diag::SecurityAccess::SecurityAccess
DiagnosticServiceValidationInterface	[SWS_DM_00771] Definition of API class ara::diag::ServiceValidation [SWS_DM_00772] Definition of API function ara::diag::ServiceValidation::ServiceValidation
DiagnosticSovdAuthorizationInterface	[SWS_DM_01490] Definition of API class ara::diag::SovdAuthorization [SWS_DM_01489] Definition of API function ara::diag::SovdAuthorization::SovdAuthorization
DiagnosticSovdBulkDataInterface	[SWS_DM_01841] Definition of API class ara::diag::SovdBulkData





Port Interface	API Class / Function
	[SWS_DM_01847] Definition of API function ara::diag::SovdBulkData::SovdBulkData
DiagnosticSovdConfigurationInterface	[SWS_DM_01860] Definition of API class ara::diag::SovdConfiguration
	[SWS_DM_01864] Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
DiagnosticSovdProximityChallengeInterface	[SWS_DM_01481] Definition of API class ara::diag::SovdProximityChallenge
	[SWS_DM_01480] Definition of API function ara::diag::SovdProximityChallenge::SovdProximityChallenge
DiagnosticSovdServiceValidationInterface	[SWS_DM_01822] Definition of API class ara::diag::SovdServiceValidation
	[SWS_DM_01823] Definition of API function ara::diag::SovdServiceValidation::SovdServiceValidation
DiagnosticSovdUpdateInterface	[SWS_DM_01874] Definition of API class ara::diag::SovdSwUpdate
	[SWS_DM_01910] Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
DiagnosticTransmitCertificateInterface	[SWS_DM_01961] Definition of API class ara::diag::TransmitCertificate
	[SWS_DM_01962] Definition of API function ara::diag::TransmitCertificate::TransmitCertificate
DiagnosticUploadInterface	[SWS_DM_00794] Definition of API class ara::diag::UploadService
	[SWS_DM_00797] Definition of API function ara::diag::UploadService::UploadService

Table 8.2: PortInterface (sub-class) to API class / function binding

This chapter lists all provided and required C++ API interfaces of the DM.

8.2 Diagnostic Communication-related APIs

8.3 Header: ara/diag/authentication.h

8.3.1 Class: Authentication

[SWS_DM_01123] Definition of API class ara::diag::Authentication

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	class
Port Interfaces:	DiagnosticAuthenticationInterface
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.

]

8.3.1.1 Public Member Functions

8.3.1.1.1 Special Member Functions

8.3.1.1.1.1 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:	<code>class ara::diag::Authentication</code>
Syntax:	<code>Authentication (Authentication &&) noexcept=delete;</code>
Description:	Move constructor of Authentication.

]

8.3.1.1.1.2 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:	<code>class ara::diag::Authentication</code>
Syntax:	<code>Authentication & operator= (Authentication &&)=delete;</code>
Description:	Move assignment operator of Authentication.

]

8.3.1.1.1.3 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:	<code>class ara::diag::Authentication</code>
Syntax:	<code>virtual ~Authentication () noexcept;</code>
Exception Safety:	exception safe





Thread Safety:	not thread-safe
Description:	Destructor of Authentication.

]

8.3.1.1.2 Constructors

8.3.1.1.2.1 Authentication

[SWS_DM_01124] Definition of API function `ara::diag::Authentication::Authentication`

Upstream requirements: [RS_Diag_04251](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/authentication.h"	
Scope:	<code>class ara::diag::Authentication</code>	
Syntax:	explicit Authentication (const ara::core::InstanceSpecifier &specifier, <code>ConcurrencyType</code> concurrencyType) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a <code>PortPrototype</code> of a <code>DiagnosticAuthenticationInterface</code> .
	concurrencyType	Specifies if the interface is implemented as thread safe (<code>kConcurrent</code>) or non-thread safe (<code>kNotConcurrent</code>).
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	<code>InstanceSpecifierMappingIntegrityViolation</code>	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.
	<code>PortInterfaceMappingViolation</code>	A <code>PortPrototype</code> that is referenced by a <code>DiagnosticAuthenticationPortMapping</code> needs to be typed by a <code>DiagnosticAuthenticationInterface</code> .
	<code>ProcessMappingViolation</code>	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.
	<code>InstanceSpecifierAlreadyInUseViolation</code>	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process.
Description:	Constructor of Authentication.	

]

8.3.1.1.2.2 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:	<code>class ara::diag::Authentication</code>
Syntax:	<code>Authentication (Authentication &)=delete;</code>
Description:	Authentication shall be a single not copy-able instance.

]

8.3.1.1.3 Member Functions

8.3.1.1.3.1 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:	<code>class ara::diag::Authentication</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.3.1.1.3.2 StopOffer

[SWS_DM_01131] Definition of API function ara::diag::Authentication::StopOfferUpstream requirements: [RS_Diag_04251](#)

[

Kind:	function
Header file:	#include "ara/diag/authentication.h"
Scope:	<code>class ara::diag::Authentication</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.3.1.1.3.3 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:	<code>class ara::diag::Authentication</code>	
Syntax:	<pre>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;</pre>	
Parameters (in):	communication Configuration	As defined in ISO 14229-1:2020, 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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>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>>></code>	Returns the 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 non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
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.	

]

8.3.1.1.3.4 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:	<code>class ara::diag::Authentication</code>	
Syntax:	<code>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;</code>	
Parameters (in):	communication Configuration	As defined in ISO14229-1:2020 [1], 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 [1], this parameter provides the challenge received from 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.
	metalInfo	MetalInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns TRUE in case, the current conversation has been cancelled and FALSE otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. 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 > >>	Returns 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc.	
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	

]

8.3.1.1.3.5 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:	<pre>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;</pre>	
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.
	clientEphemeralPublicKey	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 is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
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 .	

8.3.1.1.3.6 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.

8.4 Header: ara/diag/cancellation_handler.h

8.4.1 Class: CancellationHandler

[SWS_DM_00608] Definition of API class ara::diag::CancellationHandler

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#)

[

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 cancelled.

]

8.4.1.1 Public Member Types

8.4.1.1.1 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 cancelled in DM.

]

8.4.1.2 Public Member Functions

8.4.1.2.1 Special Member Functions

8.4.1.2.1.1 Copy Constructor

[SWS_DM_00611] Definition of API function `ara::diag::CancellationHandler::CancellationHandler`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#), [RS_AP_00145](#)

[

Kind:	function
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	<code>class ara::diag::CancellationHandler</code>
Syntax:	<code>CancellationHandler (const CancellationHandler &)=delete;</code>
Description:	CancellationHandler shall be a single not copy-able instance.

]

8.4.1.2.1.2 Default Constructor

[SWS_DM_00609] Definition of API function `ara::diag::CancellationHandler::CancellationHandler`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#), [RS_AP_00146](#)

[

Kind:	function
Header file:	#include "ara/diag/cancellation_handler.h"
Scope:	<code>class ara::diag::CancellationHandler</code>
Syntax:	<code>CancellationHandler ()=delete;</code>
Description:	CancellationHandler shall not be constructable by applications but the DM core only.

]

8.4.1.2.1.3 Move Constructor

[SWS_DM_00610] Definition of API function `ara::diag::CancellationHandler::CancellationHandler`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#), [RS_AP_00145](#)

[

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	<code>class ara::diag::CancellationHandler</code>	
Syntax:	<code>CancellationHandler (CancellationHandler &&other) noexcept=default;</code>	
Parameters (in):	other	Object to move-construct from.
Exception Safety:	exception safe	
Description:	Move constructor of CancellationHandler.	

]

8.4.1.2.1.4 Move Assignment Operator

[SWS_DM_00612] Definition of API function `ara::diag::CancellationHandler::operator=`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#), [RS_AP_00145](#)

[

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	<code>class ara::diag::CancellationHandler</code>	
Syntax:	<code>CancellationHandler & operator= (CancellationHandler &&other) noexcept=default;</code>	
Parameters (in):	other	Object to move-assign from
Return value:	CancellationHandler &	CancellationHandler & Reference to self.
Exception Safety:	exception safe	
Description:	Move assignment operator of CancellationHandler.	

]

8.4.1.2.2 Member Functions

8.4.1.2.2.1 IsCanceled

[SWS_DM_00614] Definition of API function `ara::diag::CancellationHandler::IsCanceled`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#)

[

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	<code>class ara::diag::CancellationHandler</code>	
Syntax:	<code>bool IsCanceled () const noexcept;</code>	
Return value:	bool	bool Returns true, if the diagnostic service execution is cancelled in DM.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Reports whether the diagnostic service execution is cancelled in DM.	

]

8.4.1.2.2.2 SetNotifier

[SWS_DM_00615] Definition of API function `ara::diag::CancellationHandler::SetNotifier`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#)

[

Kind:	function	
Header file:	#include "ara/diag/cancellation_handler.h"	
Scope:	<code>class ara::diag::CancellationHandler</code>	
Syntax:	<code>void SetNotifier (CancellationHandlerSetNotifier notifier) noexcept;</code>	
Parameters (in):	notifier	Notification function that is called upon diagnostic service execution is cancelled 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 cancelled in DM. A consecutive call of this method will overwrite the previous registered notifier.	

]

8.4.1.2.2.3 operator=

[SWS_DM_00613] Definition of API function ara::diag::CancellationHandler::operator=

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04277](#), [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.

]

8.5 Header: ara/diag/client_authentication.h

8.5.1 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.

]

8.5.1.1 Public Member Types

8.5.1.1.1 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:	<code>class ara::diag::ClientAuthentication</code>
Symbol:	ClientAuthenticationSetNotifier
Syntax:	<code>using ClientAuthenticationSetNotifier = std::function<void(DiagnosticAuthState)>;</code>
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 <code>kDeAuthenticated</code> State occurred due to an S3 timeout.

]

8.5.1.1.2 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:	<code>class ara::diag::ClientAuthentication</code>
Symbol:	DiagnosticAuthRole
Syntax:	<code>using DiagnosticAuthRole = ara::core::String;</code>
Description:	The supported values for the Diagnostic Authentication roles are specified in the Diagnostic Extract.

]

8.5.1.1.3 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:	<code>class ara::diag::ClientAuthentication</code>	
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 authenticated.
Description:	Possible values of the Authentication State of the client.	

]

8.5.1.2 Public Member Functions

8.5.1.2.1 Special Member Functions

8.5.1.2.1.1 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:	<code>class ara::diag::ClientAuthentication</code>	
Syntax:	<code>ClientAuthentication (ClientAuthentication &&other) noexcept=default;</code>	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Description:	Move constructor of ClientAuthentication.	

]

8.5.1.2.1.2 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:	<code>class ara::diag::ClientAuthentication</code>
Syntax:	<code>~ClientAuthentication () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ClientAuthentication.

]

8.5.1.2.2 Constructors

8.5.1.2.2.1 ClientAuthentication

[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:	<code>class ara::diag::ClientAuthentication</code>
Syntax:	<code>ClientAuthentication (ClientAuthentication const &other)=delete;</code>
Description:	Copy constructor of ClientAuthentication cannot be used.

]

8.5.1.2.3 Member Functions

8.5.1.2.3.1 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:	<code>class ara::diag::ClientAuthentication</code>	
Syntax:	<code>ara::core::Result< ClientAuthenticationHandle > Authenticate (ara::core::Vector< DiagnosticAuthRole > userRoles) noexcept;</code>	
Parameters (in):	userRoles	The user roles to set on the diagnostic client
Return value:	<code>ara::core::Result< ClientAuthenticationHandle ></code>	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:	DiagErrc::kServiceNot Available	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.	

8.5.1.2.3.2 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:	<code>class ara::diag::ClientAuthentication</code>	
Syntax:	<code>ara::core::Result< DiagnosticAuthState > GetState () const noexcept;</code>	
Return value:	<code>ara::core::Result< DiagnosticAuthState ></code>	The Authentication State of the Diagnostic Client or error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.

]

8.5.1.2.3.3 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:	<code>class ara::diag::ClientAuthentication</code>	
Syntax:	<pre>ara::core::Result< ClientAuthenticationHandle > OverrideDefaultRoles (ara::core::Vector< DiagnosticAuthRole > defaultRoles, std::chrono::milliseconds timeout) noexcept;</pre>	
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< Client AuthenticationHandle >	Operation result
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.5.1.2.3.4 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:	<code>class ara::diag::ClientAuthentication</code>	





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 >	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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, 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.	

]

8.5.1.2.3.5 operator=(ClientAuthentication&&) &

[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	ClientAuthentication& Reference to self.
Exception Safety:	exception safe	
Description:	Move assignment operator of ClientAuthentication.	

]

8.5.1.2.3.6 operator=(ClientAuthentication const&)

[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:	<code>auto operator= (ClientAuthentication const &other) -> ClientAuthentication &=delete;</code>
Description:	Copy assignment operator of ClientAuthentication cannot be used.

]

8.6 Header: ara/diag/client_authentication_handle.h

8.6.1 Class: ClientAuthenticationHandle

[SWS_DM_01145] Definition of API class ara::diag::ClientAuthenticationHandle

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	class
Header file:	<code>#include "ara/diag/client_authentication_handle.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace ara::diag
Symbol:	ClientAuthenticationHandle
Syntax:	<code>class ClientAuthenticationHandle final {...};</code>
Description:	Definition of the ClientAuthenticationHandle which is returned to the application when an AuthenticationState is set by the application.

]

8.6.1.1 Public Member Functions

8.6.1.1.1 Special Member Functions

8.6.1.1.1.1 Default Constructor

[SWS_DM_01146] Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/client_authentication_handle.h"</code>
Scope:	<code>class ara::diag::ClientAuthenticationHandle</code>
Syntax:	<code>ClientAuthenticationHandle () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructor of ClientAuthenticationHandle.

]

8.6.1.1.1.2 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:	<code>class ara::diag::ClientAuthenticationHandle</code>
Syntax:	<code>ClientAuthenticationHandle (ClientAuthenticationHandle &&other) noexcept=default;</code>
Parameters (in):	other Object to move-construct from
Exception Safety:	exception safe
Description:	Move constructor of ClientAuthenticationHandle.
]	

8.6.1.1.1.3 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:	<code>class ara::diag::ClientAuthenticationHandle</code>
Syntax:	<code>~ClientAuthenticationHandle () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ClientAuthenticationHandle.
]	

8.6.1.1.2 Constructors

8.6.1.1.2.1 ClientAuthenticationHandle

[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:	<code>class ara::diag::ClientAuthenticationHandle</code>
Syntax:	<code>ClientAuthenticationHandle (ClientAuthenticationHandle const &other)=delete;</code>
Description:	Copy constructor of ClientAuthenticationHandle cannot be used.

]

8.6.1.1.3 Member Functions

8.6.1.1.3.1 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:	<code>class ara::diag::ClientAuthenticationHandle</code>	
Syntax:	<code>ara::core::Result< void > Append (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;</code>	
Parameters (in):	dynamicAccessList	The DynamicAccessList to be appended in the client.
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.6.1.1.3.2 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:	<code>ara::core::Result< void > Refresh () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

8.6.1.1.3.3 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:	<code>ara::core::Result< void > Revoke () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

8.6.1.1.3.4 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:	<code>class ara::diag::ClientAuthenticationHandle</code>	
Syntax:	<code>ara::core::Result< void > Set (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;</code>	
Parameters (in):	dynamicAccessList	The new DynamicAccessList to be set in the client.
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.6.1.1.3.5 `operator=(ClientAuthenticationHandle&&) &`

[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:	<code>class ara::diag::ClientAuthenticationHandle</code>	
Syntax:	<code>auto operator= (ClientAuthenticationHandle &&other) & noexcept -> ClientAuthenticationHandle & =default;</code>	
Parameters (in):	other	Object to move-assign from.
Return value:	ClientAuthentication Handle	ClientAuthenticationHandle& Reference to self.
Exception Safety:	exception safe	
Description:	Move assignment operator of ClientAuthenticationHandle.	

]

8.6.1.1.3.6 operator=(ClientAuthenticationHandle const&)

[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.

]

8.7 Header: ara/diag/communication_control.h

8.7.1 Class: CommunicationControl

[SWS_DM_00804] Definition of API class ara::diag::CommunicationControl

Upstream requirements: [RS_Diag_04278](#)

[

Kind:	class
Port Interfaces:	DiagnosticComControlInterface
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:	0x28 CommunicationControl interface

]

8.7.1.1 Public Member Functions

8.7.1.1.1 Special Member Functions

8.7.1.1.1.1 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:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>CommunicationControl (CommunicationControl &&) noexcept=delete;</code>
Description:	Move constructor of CommunicationControl.

]

8.7.1.1.1.2 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:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>CommunicationControl & operator= (CommunicationControl &&)=delete;</code>
Description:	Move assignment operator of CommunicationControl.

]

8.7.1.1.1.3 Destructor

[SWS_DM_00807] Definition of API function `ara::diag::CommunicationControl::~~CommunicationControl`

Upstream requirements: [RS_AP_00134](#), [RS_Diag_04278](#)

[

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>virtual ~CommunicationControl () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class CommunicationControl.

]

8.7.1.1.2 Constructors

8.7.1.1.2.1 CommunicationControl

[SWS_DM_00806] Definition of API function `ara::diag::CommunicationControl::CommunicationControl`

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04278](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	<code>class ara::diag::CommunicationControl</code>	
Syntax:	<code>explicit CommunicationControl (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticComControlInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent)..
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.
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticComControlInterface needs to be referenced by a DiagnosticServiceGenericMapping .
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.





	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.
Description:	Constructor of CommunicationControl.	

]

8.7.1.1.2.2 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:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>CommunicationControl (CommunicationControl &)=delete;</code>
Description:	CommunicationControl shall be a single not copy-able instance.

]

8.7.1.1.3 Member Functions

8.7.1.1.3.1 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_04278](#)

[

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	<code>class ara::diag::CommunicationControl</code>	
Syntax:	<code>virtual ara::core::Future< void > CommCtrlRequest (ComCtrlRequest ParamsType controlType, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</code>	
Parameters (in):	controlType	All UDS request parameters packed into a structure since it holds optional elements
	metaInfo	MetaInfo of the request.





	cancellationHandler	This parameter is used to • Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. • Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for CommunicationControl (0x28) with any subfunction as subfunction value is part of the argument list.	

]

8.7.1.1.3.2 Offer

[SWS_DM_00809] Definition of API function `ara::diag::CommunicationControl::Offer`

Upstream requirements: [RS_AP_00119](#), [RS_AP_00139](#), [RS_Diag_04278](#)

[

Kind:	function	
Header file:	#include "ara/diag/communication_control.h"	
Scope:	<code>class ara::diag::CommunicationControl</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.7.1.1.3.3 StopOffer

[SWS_DM_00810] Definition of API function ara::diag::CommunicationControl::StopOffer

Upstream requirements: [RS_Diag_04278](#)

[

Kind:	function
Header file:	#include "ara/diag/communication_control.h"
Scope:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.7.1.1.3.4 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:	<code>class ara::diag::CommunicationControl</code>
Syntax:	<code>CommunicationControl & operator= (CommunicationControl &)=delete;</code>
Description:	CommunicationControl shall be a single not assignable instance.

]

8.7.2 Struct: ComCtrlRequestParamsType

[SWS_DM_00805] Definition of API class ara::diag::CommunicationControl::ComCtrlRequestParamsType

Upstream requirements: [RS_Diag_04278](#)

[

Kind:	struct
Header file:	#include "ara/diag/communication_control.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"





Scope:	<code>class ara::diag::CommunicationControl</code>
Symbol:	<code>ComCtrlRequestParamsType</code>
Syntax:	<code>struct ComCtrlRequestParamsType { ... };</code>
Description:	ComCtrlRequestParamsType is a structure, which holds all parameters of an UDS 0x28 communicationControl request.

]

8.7.2.1 Public Member Variables

8.7.2.1.1 communicationType

[SWS_DM_01563] Definition of API variable `ara::diag::CommunicationControl::ComCtrlRequestParamsType::communicationType`

Upstream requirements: [RS_Diag_04278](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/communication_control.h"</code>
Scope:	<code>struct ara::diag::CommunicationControl::ComCtrlRequestParamsType</code>
Symbol:	<code>communicationType</code>
Type:	<code>std::uint8_t</code>
Syntax:	<code>std::uint8_t communicationType;</code>
Description:	CommunicationType from UDS request.

]

8.7.2.1.2 controlType

[SWS_DM_01562] Definition of API variable `ara::diag::CommunicationControl::ComCtrlRequestParamsType::controlType`

Upstream requirements: [RS_Diag_04278](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/communication_control.h"</code>
Scope:	<code>struct ara::diag::CommunicationControl::ComCtrlRequestParamsType</code>
Symbol:	<code>controlType</code>
Type:	<code>std::uint8_t</code>
Syntax:	<code>std::uint8_t controlType;</code>
Description:	ControlType from UDS request.

]

8.7.2.1.3 nodeIdentificationNumber

[SWS_DM_01564] Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::nodeIdentificationNumber

Upstream requirements: [RS_Diag_04278](#)

[

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.

]

8.8 Header: ara/diag/concurrency.h

8.8.1 Non-Member Types

8.8.1.1 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.	

]

8.8.2 Struct: DataIdentifierConcurrencyType

[SWS_DM_00936] Definition of API class ara::diag::DataIdentifierConcurrencyType

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.

]

8.8.2.1 Public Member Variables

8.8.2.1.1 read

[SWS_DM_00937] Definition of API variable ara::diag::DataIdentifierConcurrencyType::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 {ConcurrencyType::kNotConcurrent};
Description:	Concurrency type for Reads.

]

8.8.2.1.2 readWrite

[SWS_DM_00939] Definition of API variable ara::diag::DataIdentifierConcurrencyType::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 {ConcurrencyType::kNotConcurrent};
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.

]

8.8.2.1.3 write

[SWS_DM_00938] Definition of API variable ara::diag::DataIdentifierConcurrencyType::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 {ConcurrencyType::kNotConcurrent};
Description:	Concurrency type for Writes.

]

8.8.3 Struct: SovdContentConcurrencyType

[SWS_DM_02182] Definition of API class ara::diag::SovdContentConcurrencyType

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:	SovdContentConcurrencyType
Syntax:	struct SovdContentConcurrencyType {...};
Description:	Specifies the Concurrency type of a SOVD Content related port.

]

8.8.3.1 Public Member Variables

8.8.3.1.1 read

[SWS_DM_02183] Definition of API variable ara::diag::SovdContentConcurrencyType::read

Upstream requirements: [RS_Diag_04166](#)

[

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	struct ara::diag::SovdContentConcurrencyType
Symbol:	read
Type:	ConcurrencyType
Syntax:	ConcurrencyType read {ConcurrencyType::kNotConcurrent};
Description:	Concurrency type for Reads.

]

8.8.3.1.2 readWrite

[SWS_DM_02185] Definition of API variable ara::diag::SovdContentConcurrencyType::readWrite

Upstream requirements: [RS_Diag_04166](#)

[

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	<code>struct ara::diag::SovdContentConcurrencyType</code>
Symbol:	readWrite
Type:	ConcurrencyType
Syntax:	<code>ConcurrencyType readWrite {ConcurrencyType::kNotConcurrent};</code>
Description:	Concurrency type for calling Read and Write methods in a concurrent way. If set to <code>kConcurrent</code> the DM can call Read and Write APIs concurrently.

]

8.8.3.1.3 write

[SWS_DM_02184] Definition of API variable ara::diag::SovdContentConcurrencyType::write

Upstream requirements: [RS_Diag_04166](#)

[

Kind:	variable
Header file:	#include "ara/diag/concurrency.h"
Scope:	<code>struct ara::diag::SovdContentConcurrencyType</code>
Symbol:	write
Type:	ConcurrencyType
Syntax:	<code>ConcurrencyType write {ConcurrencyType::kNotConcurrent};</code>
Description:	Concurrency type for Writes.

]

8.9 Header: ara/diag/conversation.h

8.9.1 Non-Member Types

8.9.1.1 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.	

]

8.9.1.2 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. .	

]

8.9.1.3 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:	<code>using SessionControlType = std::uint8_t;</code>
Description:	Type for the active diagnostic session. .

]

8.9.2 Global Variables

8.9.2.1 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:	<code>constexpr SessionControlType kDefaultSession = 0x01;</code>
Description:	Default session according to ISO 14229-1.

]

8.9.2.2 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 ISO 14229-1.

]

8.9.2.3 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.

]

8.9.2.4 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 ISO 14229-1.

]

8.9.2.5 kSafetySystemDiagnosticSession

[SWS_DM_01281] Definition of API variable ara::diag::kSafetySystemDiagnosticSession

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 ISO 14229-1.

]

8.9.3 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.

]

8.9.3.1 Public Member Types

8.9.3.1.1 Type Alias: ActivityStatusTypeNotifier

[SWS_DM_02126] Definition of API type ara::diag::Conversation::ActivityStatusTypeNotifier

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#)

[

Kind:	type alias
Header file:	#include "ara/diag/conversation.h"
Scope:	<code>class ara::diag::Conversation</code>
Symbol:	ActivityStatusTypeNotifier
Syntax:	<code>using ActivityStatusTypeNotifier = std::function<void(ActivityStatusType)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the activity is changed.

]

8.9.3.1.2 Type Alias: SecurityLevelTypeNotifier

[SWS_DM_02128] Definition of API type ara::diag::Conversation::SecurityLevelTypeNotifier

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#)

[

Kind:	type alias
Header file:	#include "ara/diag/conversation.h"
Scope:	<code>class ara::diag::Conversation</code>
Symbol:	SecurityLevelTypeNotifier
Syntax:	<code>using SecurityLevelTypeNotifier = std::function<void(SecurityLevelType)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the SecurityLevel is changed.

]

8.9.3.1.3 Type Alias: SessionControlTypeNotifier

[SWS_DM_02127] Definition of API type `ara::diag::Conversation::SessionControlTypeNotifier`

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#)

Kind:	type alias
Header file:	<code>#include "ara/diag/conversation.h"</code>
Scope:	<code>class ara::diag::Conversation</code>
Symbol:	<code>SessionControlTypeNotifier</code>
Syntax:	<code>using SessionControlTypeNotifier = std::function<void(SessionControlType)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the Session is changed.

8.9.3.2 Public Member Functions

8.9.3.2.1 Special Member Functions

8.9.3.2.1.1 Move Constructor

[SWS_DM_02092] Definition of API function `ara::diag::Conversation::Conversation`

Status: DRAFT

Upstream requirements: [RS_AP_00134](#)

Kind:	function
Header file:	<code>#include "ara/diag/conversation.h"</code>
Scope:	<code>class ara::diag::Conversation</code>
Syntax:	<code>Conversation (Conversation &&)=delete;</code>
Description:	Move constructor of conversation is deleted.

8.9.3.2.1.2 Copy Constructor

[SWS_DM_02090] Definition of API function ara::diag::Conversation::Conversation

Status: DRAFT

Upstream requirements: [RS_AP_00134](#)

[

Kind:	function
Header file:	#include "ara/diag/conversation.h"
Scope:	<code>class ara::diag::Conversation</code>
Syntax:	<code>Conversation (const Conversation &)=delete;</code>
Description:	Copy constructor of conversation is deleted.

]

8.9.3.2.1.3 Copy Assignment Operator

[SWS_DM_02091] Definition of API function ara::diag::Conversation::operator=

Status: DRAFT

Upstream requirements: [RS_AP_00134](#)

[

Kind:	function
Header file:	#include "ara/diag/conversation.h"
Scope:	<code>class ara::diag::Conversation</code>
Syntax:	<code>Conversation & operator= (const Conversation &)=delete;</code>
Description:	Copy assignment operator of conversation is deleted.

]

8.9.3.2.1.4 Move Assignment Operator

[SWS_DM_02093] Definition of API function ara::diag::Conversation::operator=

Status: DRAFT

Upstream requirements: [RS_AP_00134](#)

[

Kind:	function
Header file:	#include "ara/diag/conversation.h"
Scope:	<code>class ara::diag::Conversation</code>
Syntax:	<code>Conversation & operator= (Conversation &&)=delete;</code>
Description:	Move assignment operator of conversation is deleted.

]

8.9.3.2.2 Member Functions

8.9.3.2.2.1 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< ActivityStatusType > GetActivityStatus () noexcept;</code>	
Return value:	<code>ara::core::Result< ActivityStatusType ></code>	the activity status of the conversation
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Represents the status of an active conversation.	

8.9.3.2.2.2 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>static ara::core::Vector< std::reference_wrapper< Conversation > > GetAllConversations () noexcept;</code>	
Return value:	<code>ara::core::Vector< std::reference_wrapper< Conversation > ></code>	A Vector of all possible Conversation objects
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get all possible conversations.	

8.9.3.2.2.3 GetConversation

[SWS_DM_00692] Definition of API function ara::diag::Conversation::GetConversationUpstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04170](#)

[

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>static ara::core::Result< Conversation & > GetConversation (const MetaInfo &metaInfo) noexcept;</code>	
Parameters (in):	metaInfo	MetaInfo of the request.
Return value:	ara::core::Result< Conversation & >	Conversation object or error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get one conversation based on given MetaInfo.	

]

8.9.3.2.2.4 GetConversationIdentifier

[SWS_DM_00700] Definition of API function ara::diag::Conversation::GetConversationIdentifierUpstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#)

[

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< ConversationIdentifierType > GetConversationIdentifier () noexcept;</code>	
Return value:	ara::core::Result< ConversationIdentifierType >	Returns the conversation information
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.9.3.2.2.5 GetCurrentActiveConversations

[SWS_DM_00783] Definition of API function ara::diag::Conversation::GetCurrentActiveConversationsUpstream requirements: [RS_Diag_04166](#), [RS_Diag_04169](#), [RS_Diag_04209](#)

[

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>static ara::core::Vector< std::reference_wrapper< Conversation > > GetCurrentActiveConversations () noexcept;</code>	
Return value:	<code>ara::core::Vector< std::reference_wrapper< Conversation > ></code>	A Vector of all currently active (<code>GetActivityStatus() == kActive</code>) Conversation objects.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get all currently active conversations.	

]

8.9.3.2.2.6 GetDiagnosticSecurityLevel

[SWS_DM_00698] Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevelUpstream requirements: [RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#)

[

Kind:	function	
Header file:	#include "ara/diag/conversation.h"	
Scope:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< SecurityLevelType > GetDiagnosticSecurityLevel () noexcept;</code>	
Return value:	<code>ara::core::Result< SecurityLevelType ></code>	Returns the current SecurityLevel
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.9.3.2.2.7 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< ara::core::StringView > GetDiagnosticSecurityLevelShortName (SecurityLevelType securityLevel) noexcept;</code>	
Parameters (in):	securityLevel	Security level enum the shortname shall be returned for.
Return value:	<code>ara::core::Result< ara::core::StringView ></code>	<code>ara::core::Result<ara::core::StringView></code> the SecurityLevel as short Name; <code>DiagnosticSecurityLevel.shortName</code> .
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Converts the given diagnostic SecurityLevel into the ShortName.	

8.9.3.2.2.8 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< SessionControlType > GetDiagnosticSession () noexcept;</code>	
Return value:	<code>ara::core::Result< SessionControlType ></code>	the current session
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	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.	

8.9.3.2.2.9 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< ara::core::StringView > GetDiagnosticSessionShortName (SessionControlType session) noexcept;</code>	
Parameters (in):	session	Diagnostic session the shortname shall be returned for.
Return value:	<code>ara::core::Result< ara::core::StringView ></code>	<code>ara::core::Result<ara::core::StringView></code> the session as short Name; <code>DiagnosticSession.shortName</code> .
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Converts the given diagnostic session into the ShortName.	

8.9.3.2.2.10 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< void > ResetToDefaultSession () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Method to reset the current session to default session.	

8.9.3.2.2.11 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 (ActivityStatusType Notifier notifier) noexcept;	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.9.3.2.2.12 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 (SessionControl TypeNotifier notifier) noexcept;	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.9.3.2.2.13 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:	<code>class ara::diag::Conversation</code>	
Syntax:	<code>ara::core::Result< void > SetSecurityLevelNotifier (SecurityLevelType Notifier notifier) noexcept;</code>	
Parameters (in):	notifier	notifier function to be called
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.9.4 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:	<code>class ara::diag::Conversation</code>
Symbol:	ConversationIdentifierType
Syntax:	<code>struct ConversationIdentifierType {...};</code>
Description:	Properties allowing an identification of the conversation.

]

8.10 Header: ara/diag/data_transfer.h

8.10.1 Non-Member Types

8.10.1.1 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.	

]

8.10.2 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.	

]

8.10.2.1 Public Member Functions

8.10.2.1.1 Special Member Functions

8.10.2.1.1.1 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:	<code>DataTransferReadByPullHandler (DataTransferReadByPullHandler &&other) noexcept=default;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Description:	Move constructs an instance of this class.	

]

8.10.2.1.1.2 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:	<code>virtual ~DataTransferReadByPullHandler () noexcept=default;</code>	
Exception Safety:	exception safe	
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before <code>ExitRead()</code> is called.	

]

8.10.2.1.2 Constructors

8.10.2.1.2.1 DataTransferReadByPullHandler

[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 copyable since only one way usage i.e. per single file transfer session.

]

8.10.2.1.3 Member Functions

8.10.2.1.3.1 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	MetaInfo of the request.





	cancellation_handler	This parameter is used to • Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. • Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< ara::core::Vector< ara::core::Byte > ></code>	A Future, which either gets readied to <code>OperationOutput</code> (transfer <code>ResponseParameterRecord</code> for a positive response message) or readied with <code>UDS ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message). Data in <code>OperationOutput.response_data</code> will be placed after <code>SID</code> as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Exits an ongoing data transfer session.	

]

8.10.2.1.3.2 Read

[SWS_DM_01513] Definition of API function `ara::diag::DataTransferReadByPullHandler::Read`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	<code>class ara::diag::DataTransferReadByPullHandler</code>	
Syntax:	<pre>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;</pre>	
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	MetaInfo of the request.
	cancellation_handler	This parameter is used to • Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. • Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.





Return value:	ara::core::Future<std::uint32_t>	A Future, which either gets readied to actual amount of data provided in the Span (for a positive response message) or readied with UDS ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in response_data will be placed after blockSequenceCounter as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
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 ara::diag Span with up to the requested Span size.	

]

8.10.2.1.3.3 operator=(DataTransferReadByPullHandler&&) &

[SWS_DM_01511] Definition of API function ara::diag::DataTransferReadByPullHandler::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:	DataTransferReadByPullHandler	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

]

8.10.2.1.3.4 operator=(DataTransferReadByPullHandler const&)

[SWS_DM_01512] Definition of API function ara::diag::DataTransferReadByPullHandler::operator=

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	<code>class ara::diag::DataTransferReadByPullHandler</code>
Syntax:	<code>auto operator= (DataTransferReadByPullHandler const &) -> DataTransferReadByPullHandler &=delete;</code>
Description:	Handlers shall not be copy-able since only one way usage i.e. per single file transfer session.

]

8.10.2.2 Protected Member Functions

8.10.2.2.1 Special Member Functions

8.10.2.2.1.1 Default Constructor

[SWS_DM_01507] Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	<code>class ara::diag::DataTransferReadByPullHandler</code>
Syntax:	<code>DataTransferReadByPullHandler () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Constructs an instance of this class.
Visibility:	protected

]

8.10.3 Class: DataTransferReadByPushHandler

[SWS_DM_01515] Definition of API class ara::diag::DataTransferReadByPushHandler

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.

]

8.10.3.1 Public Member Functions

8.10.3.1.1 Special Member Functions

8.10.3.1.1.1 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
Description:	Move constructs an instance of this class.

]

8.10.3.1.1.2 Destructor

[SWS_DM_01519] Definition of API function `ara::diag::DataTransferReadByPushHandler::~~DataTransferReadByPushHandler`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	<code>class ara::diag::DataTransferReadByPushHandler</code>
Syntax:	<code>virtual ~DataTransferReadByPushHandler () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before <code>ExitRead()</code> is called

]

8.10.3.1.2 Constructors

8.10.3.1.2.1 DataTransferReadByPushHandler

[SWS_DM_01518] Definition of API function `ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	<code>class ara::diag::DataTransferReadByPushHandler</code>
Syntax:	<code>DataTransferReadByPushHandler (DataTransferReadByPushHandler const &)=delete;</code>
Description:	Handlers shall not be copy-able since only one way usage i.e. per single file transfer session.

]

8.10.3.1.3 Member Functions

8.10.3.1.3.1 ExitRead

[SWS_DM_01523] Definition of API function `ara::diag::DataTransferReadByPushHandler::ExitRead`

Upstream requirements: [RS_Diag_04135](#)

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	<code>class ara::diag::DataTransferReadByPushHandler</code>	
Syntax:	<pre>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;</pre>	
Parameters (in):	<code>exit_type</code>	Specifies the exit reason
	<code>transfer_request_parameter_record</code>	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.
	<code>meta_info</code>	MetaInfo of the request.
	<code>cancellation_handler</code>	This parameter is used to • Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. • Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< ara::core::Vector< ara::core::Byte > ></code>	A Future, which either gets readied to <code>OperationOutput</code> (transfer <code>ResponseParameterRecord</code> for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message). Data in <code>OperationOutput.response_data</code> will be placed after SID as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Exits an ongoing data transfer session.	

8.10.3.1.3.2 Read

[SWS_DM_01522] Definition of API function `ara::diag::DataTransferReadByPushHandler::Read`

Upstream requirements: [RS_Diag_04135](#)

Kind:	function	
Header file:	#include "ara/diag/data_transfer.h"	
Scope:	<code>class ara::diag::DataTransferReadByPushHandler</code>	
Syntax:	<pre>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;</pre>	
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	MetaInfo of the request.
	cancellation_handler	This parameter is used to - Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. - Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future<</code> <code>ara::core::Vector<</code> <code>ara::core::Byte > ></code>	A Future, which either gets readied to <code>ReadPushOutput</code> (transfer <code>ResponseParameterRecord</code> for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for a negative response message). Data in <code>ReadPushOutput.responseData</code> will be placed after block <code>SequenceCounter</code> as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
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 <code>ara::diag</code>.	

8.10.3.1.3.3 operator=(DataTransferReadByPushHandler&&) &**[SWS_DM_01520] Definition of API function ara::diag::DataTransferReadByPushHandler::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:	DataTransferReadByPushHandler	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

]

8.10.3.1.3.4 operator=(DataTransferReadByPushHandler const&)**[SWS_DM_01521] Definition of API function ara::diag::DataTransferReadByPushHandler::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 &) -> DataTransferReadByPushHandler &=delete;	
Description:	Handlers shall not be assignable since only one way usage i.e. per single file transfer session.	

]

8.10.3.2 Protected Member Functions

8.10.3.2.1 Special Member Functions

8.10.3.2.1.1 Default Constructor

[SWS_DM_01516] Definition of API function `ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	#include "ara/diag/data_transfer.h"
Scope:	<code>class ara::diag::DataTransferReadByPushHandler</code>
Syntax:	<code>DataTransferReadByPushHandler () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Constructs an instance of this class.
Visibility:	protected

]

8.10.4 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 <code>ara::diag</code>
Symbol:	<code>DataTransferReadSession</code>
Syntax:	<code>class DataTransferReadSession final {...};</code>
Description:	Encapsulates all data transfer reading variants.

]

8.10.4.1 Public Member Functions

8.10.4.1.1 Special Member Functions

8.10.4.1.1.1 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:	<code>class ara::diag::DataTransferReadSession</code>
Syntax:	<code>DataTransferReadSession ()=delete;</code>
Description:	No default construction allowed.

]

8.10.4.1.1.2 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:	<code>class ara::diag::DataTransferReadSession</code>	
Syntax:	<code>DataTransferReadSession (DataTransferReadSession &&other) noexcept=default;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Description:	Move constructs an instance of this class.	

]

8.10.4.1.1.3 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:	<code>class ara::diag::DataTransferReadSession</code>
Syntax:	<code>~DataTransferReadSession () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructs an instance of this class.

]

8.10.4.1.2 Constructors

8.10.4.1.2.1 DataTransferReadSession

[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:	<code>class ara::diag::DataTransferReadSession</code>
Syntax:	<code>DataTransferReadSession (DataTransferReadSession const &)=delete;</code>
Description:	DataTransferReadSession shall be a single not copy-able instance.

]

8.10.4.1.3 Member Functions

8.10.4.1.3.1 operator=(DataTransferReadSession const&)

[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:	<code>class ara::diag::DataTransferReadSession</code>
Syntax:	<code>auto operator= (DataTransferReadSession const &) -> DataTransferReadSession &=delete;</code>
Description:	DataTransferReadSession shall be a single not assign-able instance.

]

8.10.4.1.3.2 operator=(DataTransferReadSession&&) &

[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:	<code>class ara::diag::DataTransferReadSession</code>	
Syntax:	<code>auto operator= (DataTransferReadSession &&other) & noexcept -> DataTransferReadSession &=default;</code>	
Parameters (out):	other	The other object
Return value:	DataTransferReadSession	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

]

8.10.5 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.

]

8.10.5.1 Public Member Functions

8.10.5.1.1 Special Member Functions

8.10.5.1.1.1 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
Description:	Constructs an instance of this class.

]

8.10.5.1.1.2 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:	<code>DataTransferReadSharedDataHandler (DataTransferReadSharedDataHandler &&other) noexcept=default;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Description:	Move constructs an instance of this class.	

]

8.10.5.1.1.3 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:	<code>virtual ~DataTransferReadSharedDataHandler () noexcept=default;</code>	
Exception Safety:	exception safe	
Description:	Destructs an instance of this class. An instance of this class must not be destroyed before <code>ExitRead()</code> is called.	

]

8.10.5.1.2 Constructors

8.10.5.1.2.1 DataTransferReadSharedDataHandler

[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:	<code>class ara::diag::DataTransferReadSharedDataHandler</code>
Syntax:	<code>DataTransferReadSharedDataHandler (DataTransferReadSharedDataHandler const &)=delete;</code>
Description:	Handlers shall not be copy-able since only one way usage i.e. per single file transfer session.

]

8.10.5.1.3 Member Functions

8.10.5.1.3.1 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:	<code>class ara::diag::DataTransferReadSharedDataHandler</code>	
Syntax:	<code>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;</code>	
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 of the request.





	cancellation_handler	This parameter is used to - Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. - Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< ara::core::Vector< ara::core::Byte > ></code>	Returns a Future, which either gets readied to <code>OperationOutput</code> (<code>transferResponseParameterRecord</code> for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for a negative response message). Data in <code>OperationOutput.response_data</code> will be placed after SID as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Exits an ongoing data transfer session.	

]

8.10.5.1.3.2 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:	<code>class ara::diag::DataTransferReadSharedDataHandler</code>	
Syntax:	<pre>virtual auto Read (ReleaseHandler release_handler, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Span< ara::core::Byte > >=0;</pre>	
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 is used to - Query the current cancellation status by calling <code>cancellation_handler.IsCanceled()</code>. Returns true in case, the current conversation has been cancelled and false otherwise. - Register a notifier function by calling <code>cancellation_handler.SetNotifier()</code>. The registered notifier is called if the current conversation is cancelled.





Return value:	ara::core::Future< ara::core::Span< ara::core::Byte > >	Returns a Future, which either gets readied to ReadSharedOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in ReleaseSharedHandler.responseData will be placed after the blockSequenceCounter as transferResponseParameterRecord in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Provides the 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.	

┌

8.10.5.1.3.3 operator=(DataTransferReadSharedDataHandler&&) &

[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	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

┌

8.10.5.1.3.4 operator=(DataTransferReadSharedDataHandler const&)

[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 copy-able since only one way usage i.e. per single file transfer session.

]

8.10.6 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:	<code>class DataTransferWriteHandler {...};</code>
Description:	Handles data transfers initiated by AddFile, ResumeFile or ReplaceFile ModeOfOperation handling per requested RequestWriteFile.

]

8.10.6.1 Public Member Functions

8.10.6.1.1 Special Member Functions

8.10.6.1.1.1 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:	<code>class ara::diag::DataTransferWriteHandler</code>	
Syntax:	<code>DataTransferWriteHandler (DataTransferWriteHandler &&other) noexcept=default;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Description:	Move constructs an instance of this class.	

]

8.10.6.1.1.2 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:	<code>class ara::diag::DataTransferWriteHandler</code>	
Syntax:	<code>virtual ~DataTransferWriteHandler () noexcept;</code>	
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 <code>ExitWrite()</code> is called.	

]

8.10.6.1.2 Constructors

8.10.6.1.2.1 DataTransferWriteHandler

[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:	<code>class ara::diag::DataTransferWriteHandler</code>
Syntax:	<code>DataTransferWriteHandler (DataTransferWriteHandler const &)=delete;</code>
Description:	Handlers shall not be copyable since only one way usage i.e. per single file transfer session.

]

8.10.6.1.3 Member Functions

8.10.6.1.3.1 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:	<code>class ara::diag::DataTransferWriteHandler</code>	
Syntax:	<pre>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;</pre>	
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	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns true in case, the current conversation has been cancelled and false otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< ara::core::Vector< ara::core::Byte > ></code>	Returns a Future, which either gets readied to <code>OperationOutput</code> (transferResponseParameterRecord for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message) Data in <code>OperationOutput.response_data</code> will be placed after SID as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Exits an ongoing data transfer session.	

8.10.6.1.3.2 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:	<code>class ara::diag::DataTransferWriteHandler</code>	
Syntax:	<pre>virtual auto Write (ara::core::Span< ara::core::Byte > requestData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< void >=0;</pre>	
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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	A Future, which either is of type void or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message).





Exception Safety:	exception safe
Thread Safety:	not thread-safe
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .
Description:	Appends a data chunk into application memory.

┌

8.10.6.1.3.3 operator=(DataTransferWriteHandler&&) &

[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:	DataTransferWriteHandler	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

┌

8.10.6.1.3.4 operator=(DataTransferWriteHandler const&)

[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 copyable since only one way usage i.e. per single file transfer session.	

┌

8.10.6.2 Protected Member Functions

8.10.6.2.1 Special Member Functions

8.10.6.2.1.1 Default Constructor

[SWS_DM_01540] Definition of API function `ara::diag::DataTransferWriteHandler::DataTransferWriteHandler`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/data_transfer.h"</code>
Scope:	<code>class ara::diag::DataTransferWriteHandler</code>
Syntax:	<code>DataTransferWriteHandler () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Constructs an instance of this class.
Visibility:	protected

]

8.10.7 Class: `DataTransferWriteSession`

[SWS_DM_01555] Definition of API class `ara::diag::DataTransferWriteSession`

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	class
Header file:	<code>#include "ara/diag/data_transfer.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>DataTransferWriteSession</code>
Syntax:	<code>class DataTransferWriteSession final {...};</code>
Description:	Encapsulates all data transfer writing variants.

]

8.10.7.1 Public Member Functions

8.10.7.1.1 Special Member Functions

8.10.7.1.1.1 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:	<code>class ara::diag::DataTransferWriteSession</code>
Syntax:	<code>DataTransferWriteSession ()=delete;</code>
Description:	No default construction allowed.

]

8.10.7.1.1.2 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:	<code>class ara::diag::DataTransferWriteSession</code>	
Syntax:	<code>DataTransferWriteSession (DataTransferWriteSession &&other) noexcept=default;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Description:	Move constructs an instance of this class.	

]

8.10.7.1.1.3 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:	<code>class ara::diag::DataTransferWriteSession</code>
Syntax:	<code>~DataTransferWriteSession () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructs an instance of this class.

]

8.10.7.1.2 Constructors

8.10.7.1.2.1 DataTransferWriteSession

[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:	<code>class ara::diag::DataTransferWriteSession</code>
Syntax:	<code>DataTransferWriteSession (DataTransferWriteSession const &)=delete;</code>
Description:	DataTransferWriteSession shall be a single not copy-able instance.

]

8.10.7.1.3 Member Functions

8.10.7.1.3.1 operator=(DataTransferWriteSession const&)

[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:	<code>class ara::diag::DataTransferWriteSession</code>
Syntax:	<code>auto operator= (DataTransferWriteSession const &) -> DataTransferWriteSession &=delete;</code>
Description:	DataTransferWriteSession shall be a single not assignable instance.

]

8.10.7.1.3.2 operator=(DataTransferWriteSession&&) &

[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:	<code>class ara::diag::DataTransferWriteSession</code>	
Syntax:	<code>auto operator= (DataTransferWriteSession &&other) & noexcept -> DataTransferWriteSession &=default;</code>	
Parameters (out):	other	The other object
Return value:	DataTransferWriteSession	Reference to self
Exception Safety:	exception safe	
Description:	Move assigns an instance of this class.	

]

8.11 Header: ara/diag/diag_error_domain.h

8.11.1 Non-Member Types

8.11.1.1 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 .
	kConfigurationMismatch	= 102 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 exits for the requested event.
	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. .
	kCustomResetTypeNotSupported	= 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.	

]

8.11.1.2 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.
	kConfigurationMismatch	= 102 Configuration does not match DEXT.
	kDebouncingConfigurationInconsistent	= 103 monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
Description:	The DiagOfferErrc enumeration defines the error codes for the DiagOfferErrorDomain.	

]

8.11.1.3 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.
	kConfigurationMismatch	= 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 exits for the requested event.
	Description:	The DiagReportingErrc enumeration defines the error codes for the DiagReportingErrorDomain.

8.11.2 Non-Member Functions

8.11.2.1 Other

8.11.2.1.1 GetDiagDomain

[SWS_DM_00524] Definition of API function ara::diag::GetDiagDomain

Upstream requirements: [RS_AP_00119](#)

[	
Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	namespace ara::diag





Syntax:	<code>constexpr const ara::core::ErrorDomain & GetDiagDomain () noexcept;</code>	
Return value:	<code>const ara::core::ErrorDomain &</code>	reference to the DiagErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagErrorDomain instance.	

]

8.11.2.1.2 GetDiagOfferDomain

[SWS_DM_01003] Definition of API function `ara::diag::GetDiagOfferDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function	
Header file:	<code>#include "ara/diag/diag_error_domain.h"</code>	
Scope:	namespace <code>ara::diag</code>	
Syntax:	<code>constexpr const ara::core::ErrorDomain & GetDiagOfferDomain () noexcept;</code>	
Return value:	<code>const ara::core::ErrorDomain &</code>	reference to the DiagOfferErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagErrorDomain instance.	

]

8.11.2.1.3 GetDiagReportingDomain

[SWS_DM_01004] Definition of API function `ara::diag::GetDiagReportingDomain`

Upstream requirements: [RS_AP_00119](#), [RS_AP_00125](#)

[

Kind:	function	
Header file:	<code>#include "ara/diag/diag_error_domain.h"</code>	
Scope:	namespace <code>ara::diag</code>	
Syntax:	<code>constexpr const ara::core::ErrorDomain & GetDiagReportingDomain () noexcept;</code>	
Return value:	<code>const ara::core::ErrorDomain &</code>	reference to the DiagOfferErrorDomain instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global DiagReportingErrorDomain instance.	

]

8.11.2.1.4 MakeErrorCode

[SWS_DM_01006] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.11.2.1.5 MakeErrorCode

[SWS_DM_00525] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.11.2.1.6 MakeErrorCode

[SWS_DM_01005] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: [RS_AP_00119](#)

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.	

8.11.3 Class: DiagErrorDomain

[SWS_DM_00517] Definition of API class ara::diag::DiagErrorDomain

Upstream requirements: [RS_AP_00119](#)

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:	<pre>class DiagErrorDomain final : public ara::core::ErrorDomain {...};</pre>
Unique ID:	As per ara::diag::DiagErrorDomain in [SWS_CORE_90023]
Description:	Error domain for diagnostic errors.

8.11.3.1 Public Member Types

8.11.3.1.1 Type Alias: Errc

[SWS_DM_00518] Definition of API type `ara::diag::DiagErrorDomain::Errc`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagErrorDomain</code>
Symbol:	Errc
Syntax:	<code>using Errc = DiagErrc;</code>
Description:	Alias for the error code value enumeration.

]

8.11.3.1.2 Type Alias: Exception

[SWS_DM_00519] Definition of API type `ara::diag::DiagErrorDomain::Exception`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagErrorDomain</code>
Symbol:	Exception
Syntax:	<code>using Exception = DiagException;</code>
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_00520] Definition of API function `ara::diag::DiagErrorDomain::DiagErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagErrorDomain</code>
Syntax:	<code>constexpr DiagErrorDomain () noexcept;</code>
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_00522] Definition of API function `ara::diag::DiagErrorDomain::Message`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function		
Header file:	#include "ara/diag/diag_error_domain.h"		
Scope:	<code>class ara::diag::DiagErrorDomain</code>		
Syntax:	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>		
Parameters (in):	<table border="1"> <tr> <td>errorCode</td><td>the error code value</td></tr> </table>	errorCode	the error code value
errorCode	the error code value		
Return value:	<table border="1"> <tr> <td>const char *</td><td>the text message, never nullptr</td></tr> </table>	const char *	the text message, never nullptr
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 Name

[SWS_DM_00521] Definition of API function `ara::diag::DiagErrorDomain::Name`

Upstream requirements: [RS_AP_00119](#)

[

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 *	As per ara::diag::DiagErrorDomain in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]

8.11.3.2.3 ThrowAsException

[SWS_DM_00523] Definition of API function `ara::diag::DiagErrorDomain::ThrowAsException`

Upstream requirements: [RS_AP_00119](#)

[

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 <code>ErrorCode</code> instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given <code>ErrorCode</code> . 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: DiagException

[SWS_DM_00515] Definition of API class ara::diag::DiagException

Upstream requirements: [RS_AP_00119](#)

[

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:	<code>class DiagException : public ara::core::Exception {...};</code>
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 DiagException

[SWS_DM_00516] Definition of API function ara::diag::DiagException::DiagException

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function	
Header file:	#include "ara/diag/diag_error_domain.h"	
Scope:	class ara::diag::DiagException	
Syntax:	<code>explicit DiagException (ara::core::ErrorCode err) noexcept;</code>	
Parameters (in):	err	the ErrorCode
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct a new DiagException from an ErrorCode.	

]

8.11.5 Class: DiagOfferErrorDomain

[SWS_DM_00989] Definition of API class `ara::diag::DiagOfferErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	class
Header file:	#include "ara/diag/diag_error_domain.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>DiagOfferErrorDomain</code>
Base class:	<code>ara::core::ErrorDomain</code>
Syntax:	<pre>class DiagOfferErrorDomain final : public ara::core::ErrorDomain {...};</pre>
Unique ID:	As per ara::diag::DiagOfferErrorDomain in [SWS_CORE_90023]
Description:	Error domain for diagnostic offer errors.

]

8.11.5.1 Public Member Types

8.11.5.1.1 Type Alias: `Errc`

[SWS_DM_00990] Definition of API type `ara::diag::DiagOfferErrorDomain::Errc`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	class ara::diag::DiagOfferErrorDomain
Symbol:	<code>Errc</code>
Syntax:	<pre>using Errc = DiagOfferErrc;</pre>
Description:	Alias for the error code value enumeration.

]

8.11.5.1.2 Type Alias: Exception

[SWS_DM_00991] Definition of API type `ara::diag::DiagOfferErrorDomain::Exception`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagOfferErrorDomain</code>
Symbol:	Exception
Syntax:	<code>using Exception = DiagException;</code>
Description:	Alias for the exception base class.

]

8.11.5.2 Public Member Functions

8.11.5.2.1 Special Member Functions

8.11.5.2.1.1 Default Constructor

[SWS_DM_00992] Definition of API function `ara::diag::DiagOfferErrorDomain::DiagOfferErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagOfferErrorDomain</code>
Syntax:	<code>constexpr DiagOfferErrorDomain () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor.

]

8.11.5.2.2 Member Functions

8.11.5.2.2.1 Message

[SWS_DM_00994] Definition of API function `ara::diag::DiagOfferErrorDomain::Message`

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.11.5.2.2.2 Name

[SWS_DM_00993] Definition of API function `ara::diag::DiagOfferErrorDomain::Name`

Upstream requirements: [RS_AP_00119](#)

[

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 *	As per ara::diag::DiagOfferErrorDomain in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]

8.11.5.2.2.3 ThrowAsException

[SWS_DM_00995] Definition of API function ara::diag::DiagOfferErrorDomain::ThrowAsException

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.11.6 Class: DiagOfferException

[SWS_DM_00985] Definition of API class ara::diag::DiagOfferException

Upstream requirements: [RS_AP_00119](#)

[

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.

]

8.11.6.1 Public Member Functions

8.11.6.1.1 Constructors

8.11.6.1.1.1 DiagOfferException

[SWS_DM_00986] Definition of API function `ara::diag::DiagOfferException::DiagOfferException`

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.11.7 Class: DiagReportingErrorDomain

[SWS_DM_00996] Definition of API class `ara::diag::DiagReportingErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

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.

]

8.11.7.1 Public Member Types

8.11.7.1.1 Type Alias: Errc

[SWS_DM_00997] Definition of API type `ara::diag::DiagReportingErrorDomain::Errc`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagReportingErrorDomain</code>
Symbol:	Errc
Syntax:	<code>using Errc = DiagReportingErrc;</code>
Description:	Alias for the error code value enumeration.

]

8.11.7.1.2 Type Alias: Exception

[SWS_DM_00998] Definition of API type `ara::diag::DiagReportingErrorDomain::Exception`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagReportingErrorDomain</code>
Symbol:	Exception
Syntax:	<code>using Exception = DiagReportingException;</code>
Description:	Alias for the exception base class.

]

8.11.7.2 Public Member Functions

8.11.7.2.1 Special Member Functions

8.11.7.2.1.1 Default Constructor

[SWS_DM_00999] Definition of API function `ara::diag::DiagReportingErrorDomain::DiagReportingErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagReportingErrorDomain</code>
Syntax:	<code>constexpr DiagReportingErrorDomain () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor.

]

8.11.7.2.2 Member Functions

8.11.7.2.2.1 Message

[SWS_DM_01001] Definition of API function `ara::diag::DiagReportingErrorDomain::Message`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function		
Header file:	#include "ara/diag/diag_error_domain.h"		
Scope:	<code>class ara::diag::DiagReportingErrorDomain</code>		
Syntax:	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>		
Parameters (in):	<table border="1"> <tr> <td>errorCode</td><td>the error code value</td></tr> </table>	errorCode	the error code value
errorCode	the error code value		
Return value:	<table border="1"> <tr> <td>const char *</td><td>the text message, never nullptr</td></tr> </table>	const char *	the text message, never nullptr
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.		

]

8.11.7.2.2.2 Name

[SWS_DM_01000] Definition of API function `ara::diag::DiagReportingErrorDomain::Name`

Upstream requirements: [RS_AP_00119](#)

[

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 *	As per ara::diag::DiagReportingErrorDomain in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]

8.11.7.2.2.3 ThrowAsException

[SWS_DM_01002] Definition of API function `ara::diag::DiagReportingErrorDomain::ThrowAsException`

Upstream requirements: [RS_AP_00119](#)

[

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 <code>ErrorCode</code> instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given <code>ErrorCode</code> . As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

]

8.11.8 Class: DiagReportingException

[SWS_DM_00987] Definition of API class ara::diag::DiagReportingException

Upstream requirements: [RS_AP_00119](#)

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:	<code>class DiagReportingException : public ara::core::Exception {...};</code>
Description:	Exception type thrown by Diag classes.

8.11.8.1 Public Member Functions

8.11.8.1.1 Constructors

8.11.8.1.1.1 DiagReportingException

[SWS_DM_00988] Definition of API function ara::diag::DiagReportingException::DiagReportingException

Upstream requirements: [RS_AP_00119](#)

Kind:	function
Header file:	#include "ara/diag/diag_error_domain.h"
Scope:	<code>class ara::diag::DiagReportingException</code>
Syntax:	<code>explicit DiagReportingException (ara::core::ErrorCode err) noexcept;</code>
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/diag_uds_nrc_error_domain.h

8.12.1 Non-Member Types

8.12.1.1 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 0x10, This NRC indicates that the requested action has been rejected by the server. The generalReject response code shall only be implemented in the server if none of the negative response codes defined in this document meet the needs of the implementation. At no means shall this NRC be a general replacement for the response codes defined in this document.
	kServiceNotSupported	= 0x11 0x11, This NRC indicates that the requested action will not be taken because the server does not support the requested service. The server shall send this NRC in case the client has sent a request message with a service identifier which is unknown, not supported by the server, or is specified as a response service identifier. Therefore this negative response code is not shown in the list of negative response codes to be supported for a diagnostic service, because this negative response code is not applicable for supported services.
	kSubFunctionNot Supported	= 0x12 0x12, This NRC indicates that the requested action will not be taken because the server does not support the service specific parameters of the request message. The server shall send this NRC in case the client has sent a request message with a known and supported service identifier but with "SubFunction" which is either unknown or not supported.
	kIncorrectMessage LengthOrInvalidFormat	= 0x13 0x13, This NRC indicates that the requested action will not be taken because the length of the received request message does not match the prescribed length for the specified service or the format of the parameters do not match the prescribed format for the specified service.
	kResponseTooLong	= 0x14





		0x14, This NRC shall be reported by the server if the response to be generated exceeds the maximum number of bytes available by the underlying network layer. This could occur if the response message exceeds the maximum size allowed by the underlying transport protocol or if the response message exceeds the server buffer size allocated for that purpose. EXAMPLE This problem may occur when several DIDs at a time are requested and the combination of all DIDs in the response exceeds the limit of the underlying transport protocol.
	kBusyRepeatRequest	= 0x21 0x21, This NRC indicates that the server is temporarily too busy to perform the requested operation. In this circumstance the client shall perform repetition of the "identical request message" or "another request message". The repetition of the request shall be delayed by a time specified in the respective implementation documents. EXAMPLE In a multi-client environment the diagnostic request of one client might be blocked temporarily by an NRC 0x21 while a different client finishes a diagnostic task. If the server is able to perform the diagnostic task but needs additional time to finish the task and prepare the response, the NRC 7816 shall be used instead of NRC 0x21. This NRC is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kConditionsNotCorrect	= 0x22 0x22, This NRC indicates that the requested action will not be taken because the server prerequisite conditions are not met.
	kRequestSequenceError	= 0x24 0x24, This NRC indicates that the requested action will not be taken because the server expects a different sequence of request messages or message as sent by the client. This may occur when sequence sensitive requests are issued in the wrong order. EXAMPLE A successful SecurityAccess service specifies a sequence of requestSeed and sendKey as SubFunction in the request messages. If the sequence is sent different by the client, the server sends a negative response message with the negative response code 0x24 requestSequenceError.
	kNoResponseFromSubnetComponent	= 0x25 0x25, This NRC indicates that the server has received the request but the requested action could not be performed by the server as a subnet component which is necessary to supply the requested information did not respond within the specified time. The noResponseFromSubnetComponent negative response shall be implemented by gateways in electronic systems which contain electronic subnet components and which do not directly respond to the client's request. The gateway may receive the request for the subnet component and then request the necessary information from the subnet component. If the subnet component fails to respond, the server shall use this negative response to inform the client about the failure of the subnet component. This NRC is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kFailurePreventsExecutionOfRequestedAction	= 0x26





		0x26, This NRC indicates that the requested action will not be taken because a failure condition, identified by a DTC (with at least one DTC status bit for TestFailed, Pending, Confirmed or TestFailed SinceLastClear set to 1), has occurred and that this failure condition prevents the server from performing the requested action. This NRC can, for example, direct the technician to read DTCs in order to identify and fix the problem. Diagnostic services used to access DTCs shall not implement this NRC, because an external test tool may check for the above NRC and automatically request DTCs whenever the above NRC has been received. This NRC is in general supported by each diagnostic service (except the services mentioned above), as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kRequestOutOfRange	= 0x31 0x31, This NRC indicates that the requested action will not be taken because the server has detected that the request message contains a parameter which attempts to substitute a value beyond its range of authority (e.g. attempting to substitute a data byte of 111 when the data is only defined to 100), or which attempts to access a dataIdentifier/routineIdentifier that is not supported or not supported in active session. This NRC shall be implemented for all services, which allow the client to read data, write data or adjust functions by data in the server.
	kSecurityAccessDenied	= 0x33 0x33, This NRC indicates that the requested action will not be taken because the server's security strategy has not been satisfied by the client. The server shall send this NRC if one of the following cases occur: — the test conditions of the server are not met, — the required message sequence, e.g. DiagnosticSessionControl, securityAccess is not met, — the client has sent a request message which requires an unlocked server. Beside the mandatory use of this negative response code as specified in the applicable services within this document, this negative response code can also be used for any case where security is required and is not yet granted to perform the required service.
	kAuthenticationRequired	= 0x34 0x34, This NRC indicates that the requested service will not be taken because the client has insufficient rights based on its Authentication state. This NRC is in general supported by each diagnostic service, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kInvalidKey	= 0x35 0x35, This NRC indicates that the server has not given security access because the key sent by the client did not match with the key in the server's memory. This counts as an attempt to gain security. The server shall remain in the locked state and increment its internal securityAccessFailed counter as described in 10.4. (ISO 14229-1)
	kExceedNumberOfAttempts	= 0x36 0x36, This NRC indicates that the requested action will not be taken because the client has unsuccessfully attempted to gain security access more times than the server's security strategy will allow.





	kRequiredTimeDelayNot Expired	= 0x37 0x37, This NRC indicates that the requested action will not be taken because the client's latest attempt to gain security access was initiated before the server's required timeout period had elapsed.
	kSecureData TransmissionRequired	= 0x38 0x38, This NRC indicates that the requested service will not be taken because the requested action is required to be sent using a secured communication channel (i.e. A_MType is equal to secure {remote} diagnostics). This NRC is in general supported by each diagnostic service, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kSecureData TransmissionNotAllowed	= 0x39 0x39, This NRC indicates that this message was received using the SecuredDataTransmission (0x84) service. However, the requested action is not allowed to be sent using the SecuredData Transmission (0x84) service.
	kSecureDataVerification Failed	= 0x3A 0x3A, This NRC indicates that the message failed in the security sub-layer. For example, invalid Apar - Administrative parameter or "Signature/ Encryption Calculation" parameter, verification error of anti-replay counter or Signature/MAC, decryption or encryption error.
	kCertificateVerification FailedInvalidTimePeriod	= 0x50 0x50, Date and time of the server does not match the validity period of the Certificate.
	kCertificateVerification FailedInvalidSignature	= 0x51 0x51, Signature of the Certificate could not be verified.
	kCertificateVerification FailedInvalidChainOf Trust	= 0x52 0x52, Certificate could not be verified against stored information about the issuing authority.
	kCertificateVerification FailedInvalidType	= 0x53 0x53, Certificate does not match the current requested use case.
	kCertificateVerification FailedInvalidFormat	= 0x54 0x54, Certificate could not be evaluated because the format requirement has not been met.
	kCertificateVerification FailedInvalidContent	= 0x55 0x55, Certificate could not be verified because the content does not match.
	kCertificateVerification FailedInvalidScope	= 0x56 0x56, The scope of the Certificate does not match the contents of the server.
	kCertificateVerification FailedInvalidCertificate Revoked	= 0x57 0x57, Certificate received from client is invalid, because the server has revoked access for some reason.
	kOwnershipVerification Failed	= 0x58 0x58, Delivered Ownership does not match the provided challenge or could not verified with the own private key.
	kChallengeCalculation Failed	= 0x59 0x59, The challenge could not be calculated on the server side.
	kSettingAccessRights Failed	= 0x5A 0x5A, The server could not set the access rights.





	kSessionKeyCreationDerivationFailed	= 0x5B 0x5B, The server could not create or derive a session key.
	kConfigurationDataUsageFailed	= 0x5C 0x5C, The server could not work with the provided configuration data.
	kDeAuthenticationFailed	= 0x5D 0x5D, DeAuthentication was not successful, server could still be unprotected.
	kUploadDownloadNotAccepted	= 0x70 0x70, This NRC indicates that an attempt to upload/download to a server's memory cannot be accomplished due to some fault conditions.
	kTransferDataSuspended	= 0x71 0x71, This NRC indicates that a data transfer operation was halted due to some fault. The active transferData sequence shall be aborted.
	kGeneralProgrammingFailure	= 0x72 0x72, This NRC indicates that the server detected an error when erasing or programming a memory location in the permanent memory device (e.g. Flash Memory).
	kWrongBlockSequenceCounter	= 0x73 0x73, This NRC indicates that the server detected an error in the sequence of blockSequenceCounter values. Note that the repetition of a TransferData request message with a blockSequenceCounter equal to the one included in the previous TransferData request message shall be accepted by the server.
	kRequestCorrectlyReceivedResponsePending	= 0x78 0x78, This NRC indicates that the request message was received correctly, and that all parameters in the request message were valid (these checks can be delayed until after sending this NRC if executing the boot software), but the action to be performed is not yet completed and the server is not yet ready to receive another request. As soon as the requested service has been completed, the server shall send a positive response message or negative response message with a response code different from this. The negative response message with this NRC may be repeated by the server until the requested service is completed and the final response message is sent. This NRC might impact the application layer timing parameter values. The detailed specification shall be included in the data link specific implementation document. This NRC shall only be used in a negative response message if the server will not be able to receive further request messages from the client while completing the requested diagnostic service. When this NRC is used, the server shall always send a final response (positive or negative) independent of the suppressPosRspMsg IndicationBit value or the suppress requirement for responses with NRCs SNS, SFNS, SNSIAS, SFNSIAS and ROOR on functionally addressed requests. A typical example where this NRC may be used is when the client has sent a request message, which includes data to be programmed or erased in flash memory of the server. If the programming/erasing routine (usually executed out of RAM) is not able to support serial communication while writing to the flash





		memory the server shall send a negative response message with this response code. This NRC is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kSubFunctionNotSupportedInActiveSession	= 0x7E 0x7E, This NRC indicates that the requested action will not be taken because the server does not support the requested Sub Function in the session currently active. This NRC shall only be used when the requested SubFunction is known to be supported in another session, otherwise response code SFNS (SubFunctionNotSupported) shall be used (e.g. servers executing the boot software generally do not know which Sub Functions are supported in the application (and vice versa) and therefore may need to respond with NRC 0x12 instead). This NRC shall be supported by each diagnostic service with a Sub Function parameter, if not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kServiceNotSupportedInActiveSession	= 0x7F 0x7F, This NRC indicates that the requested action will not be taken because the server does not support the requested service in the session currently active. This NRC shall only be used when the requested service is known to be supported in another session, otherwise response code SNS (serviceNotSupported) shall be used (e.g. servers executing the boot software generally do not know which services are supported in the application (and vice versa) and therefore may need to respond with NRC 0x11 instead). This NRC is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kRpmTooHigh	= 0x81 0x81, This NRC indicates that the requested action will not be taken because the server prerequisite condition for RPM is not met (current RPM is above a programmed maximum threshold).
	kRpmTooLow	= 0x82 0x82, This NRC indicates that the requested action will not be taken because the server prerequisite condition for RPM is not met (current RPM is below a programmed minimum threshold).
	kEngineIsRunning	= 0x83 0x83, This NRC is required for those actuator tests which cannot be actuated while the Engine is running. This is different from RPM too high negative response, and shall be allowed.
	kEngineIsNotRunning	= 0x84 0x84, This NRC is required for those actuator tests which cannot be actuated unless the Engine is running. This is different from RPM too low negative response, and shall be allowed.
	kEngineRunTimeTooLow	= 0x85 0x85, This NRC indicates that a requested action will not be taken because the server prerequisite condition for engine run time is not met (current engine run time is below a programmed limit).
	kTemperatureTooHigh	= 0x86





		0x86, This NRC indicates that the requested action will not be taken because the server prerequisite condition for temperature is not met (current temperature is above a preprogrammed maximum threshold).
	kTemperatureTooLow	= 0x87
		0x87, This NRC indicates that the requested action will not be taken because the server prerequisite condition for temperature is not met (current temperature is below a preprogrammed minimum threshold).
	kVehicleSpeedTooHigh	= 0x88
		0x88, This NRC indicates that the requested action will not be taken because the server prerequisite condition for vehicle speed is not met (current vehicle speed is above a preprogrammed maximum threshold).
	kVehicleSpeedTooLow	= 0x89
		0x89, This NRC indicates that the requested action will not be taken because the server prerequisite condition for vehicle speed is not met (current vehicle speed is below a preprogrammed minimum threshold).
	kThrottlePedalTooHigh	= 0x8A
		0x8A, This NRC indicates that the requested action will not be taken because the server prerequisite condition for throttle/pedal position is not met (current TP/APP is above a preprogrammed maximum threshold).
	kThrottlePedalTooLow	= 0x8B
		0x8B, This NRC indicates that the requested action will not be taken because the server prerequisite condition for throttle/pedal position is not met (current TP/APP is below a preprogrammed minimum threshold).
	kTransmissionRangeNotInNeutral	= 0x8C
		0x8C, This NRC indicates that the requested action will not be taken because the server prerequisite condition for being in neutral is not met (current transmission range is not in neutral).
	kTransmissionRangeNotInGear	= 0x8D
		0x8D, This NRC indicates that the requested action will not be taken because the server prerequisite condition for being in gear is not met (current transmission range is not in gear).
	kBrakeSwitchNotClosed	= 0x8F
		0x8F, This NRC indicates that for safety reasons, this is required for certain tests before it begins, and shall be maintained for the entire duration of the test.
	kShifterLeverNotInPark	= 0x90
		0x90, This NRC indicates that for safety reasons, this is required for certain tests before it begins, and shall be maintained for the entire duration of the test.
	kTorqueConverterClutchLocked	= 0x91
		0x91, This NRC indicates that the requested action will not be taken because the server prerequisite condition for torque converter clutch is not met (current TCC status above a preprogrammed limit or locked).
	kVoltageTooHigh	= 0x92
		0x92, This NRC indicates that the requested action will not be taken because the server prerequisite condition for voltage at the primary pin of the server (ECU) is not met (current voltage is above a preprogrammed maximum threshold).





	kVoltageTooLow	= 0x93
		0x93, This NRC indicates that the requested action will not be taken because the server prerequisite condition for voltage at the primary pin of the server (ECU) is not met (current voltage is below a preprogrammed maximum threshold).
	kResourceTemporarilyNotAvailable	= 0x94
		0x94, This NRC indicates that the server has received the request but the requested action could not be performed by the server because an application which is necessary to supply the requested information is temporality not available. This NRC is in general supported by each diagnostic service, as not otherwise stated in the data link specific implementation document, therefore it is not listed in the list of applicable response codes of the diagnostic services.
	kNoProcessingNoResponse	= 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. The error codes described are defined in ISO 14229-1:2020 [1], Table A.1 "Negative Response Codes". In AUTOSAR, the three negative response code ranges, 0x00, 0x01-0x7F and 0x80-0xFF are respected, however, the value 0xFF is used to internally indicate, that there is no further processing and no response handling foreseen.	

]

8.12.2 Non-Member Functions

8.12.2.1 Other

8.12.2.1.1 GetDiagUdsNrcDomain

[SWS_DM_00536] Definition of API function `ara::diag::GetDiagUdsNrcDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	namespace <code>ara::diag</code>	
Syntax:	<pre>constexpr const ara::core::ErrorDomain & GetDiagUdsNrcDomain () noexcept;</pre>	
Return value:	const <code>ara::core::ErrorDomain &</code>	reference to the <code>DiagUdsNrcErrorDomain</code> instance
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Obtain the reference to the single global <code>DiagUdsNrcErrorDomain</code> instance.	

]

8.12.2.1.2 MakeErrorCode

[SWS_DM_00537] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.12.3 Class: DiagUdsNrcErrorDomain

[SWS_DM_00529] Definition of API class ara::diag::DiagUdsNrcErrorDomain

Upstream requirements: [RS_AP_00119](#)

[

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.

]

8.12.3.1 Public Member Types

8.12.3.1.1 Type Alias: Errc

[SWS_DM_00530] Definition of API type `ara::diag::DiagUdsNrcErrorDomain::Errc`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	<code>class ara::diag::DiagUdsNrcErrorDomain</code>
Symbol:	Errc
Syntax:	<code>using Errc = DiagUdsNrcErrc;</code>
Description:	Alias for the error code value enumeration.

]

8.12.3.1.2 Type Alias: Exception

[SWS_DM_00531] Definition of API type `ara::diag::DiagUdsNrcErrorDomain::Exception`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	<code>class ara::diag::DiagUdsNrcErrorDomain</code>
Symbol:	Exception
Syntax:	<code>using Exception = DiagUdsNrcException;</code>
Description:	Alias for the exception base class.

]

8.12.3.2 Public Member Functions

8.12.3.2.1 Special Member Functions

8.12.3.2.1.1 Default Constructor

[SWS_DM_00532] Definition of API function `ara::diag::DiagUdsNrcErrorDomain::DiagUdsNrcErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	<code>class ara::diag::DiagUdsNrcErrorDomain</code>
Syntax:	<code>constexpr DiagUdsNrcErrorDomain () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor .

]

8.12.3.2.2 Member Functions

8.12.3.2.2.1 Message

[SWS_DM_00534] Definition of API function `ara::diag::DiagUdsNrcErrorDomain::Message`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Scope:	<code>class ara::diag::DiagUdsNrcErrorDomain</code>	
Syntax:	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>	
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.	

]

8.12.3.2.2 Name

[SWS_DM_00533] Definition of API function `ara::diag::DiagUdsNrcErrorDomain::Name`

Upstream requirements: [RS_AP_00119](#)

[

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 *	As per ara::diag::DiagUdsNrcErrorDomain in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]

8.12.3.2.3 ThrowAsException

[SWS_DM_00535] Definition of API function `ara::diag::DiagUdsNrcErrorDomain::ThrowAsException`

Upstream requirements: [RS_AP_00119](#)

[

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 <code>ErrorCode</code> instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given <code>ErrorCode</code> . As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

]

8.12.4 Class: DiagUdsNrcException

[SWS_DM_00527] Definition of API class ara::diag::DiagUdsNrcException

Upstream requirements: [RS_AP_00119](#)

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:	<code>class DiagUdsNrcException : public ara::core::Exception {...};</code>
Description:	Exception type thrown by Diag classes.

8.12.4.1 Public Member Functions

8.12.4.1.1 Constructors

8.12.4.1.1.1 DiagUdsNrcException

[SWS_DM_00528] Definition of API function ara::diag::DiagUdsNrcException::DiagUdsNrcException

Upstream requirements: [RS_AP_00119](#)

Kind:	function
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Scope:	<code>class ara::diag::DiagUdsNrcException</code>
Syntax:	<code>explicit DiagUdsNrcException (ara::core::ErrorCode err) noexcept;</code>
Parameters (in):	err the ErrorCode
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Construct a new DiagException from an ErrorCode.

8.13 Header: ara/diag/diagnostic_service_dynamic_access_list.h

8.13.1 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:	<code>class DiagnosticServiceDynamicAccessList final {...};</code>
Description:	Definition of the DiagnosticServiceDynamicAccessList class, which is used by the application to build a DynamicAccessList.

]

8.13.1.1 Public Member Functions

8.13.1.1.1 Special Member Functions

8.13.1.1.1.1 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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>
Syntax:	<code>DiagnosticServiceDynamicAccessList () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructor of DiagnosticServiceDynamicAccessList.

]

8.13.1.1.1.2 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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList &&other) noexcept;</code>	
Parameters (in):	other	Object to move-construct from
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructor of <code>DiagnosticServiceDynamicAccessList</code> .	

]

8.13.1.1.1.3 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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>~DiagnosticServiceDynamicAccessList () noexcept;</code>	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Destructor of <code>DiagnosticServiceDynamicAccessList</code> .	

]

8.13.1.1.2 Constructors

8.13.1.1.2.1 DiagnosticServiceDynamicAccessList

[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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList const &other) noexcept;</code>	
Parameters (in):	other	Object to copy-construct from
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Copy constructor of DiagnosticServiceDynamicAccessList.	

]

8.13.1.1.3 Member Functions

8.13.1.1.3.1 MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::Byte String)

[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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte String serviceHead) noexcept -> DynamicAccessListDiagServiceBuilder;</code>	
Parameters (in):	serviceHead	A string of bytes to start the DynamicAccess pattern-match
Return value:	DynamicAccessListDiagServiceBuilder	Returns an instance of a diagnostic service pattern builder
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the AA to construct a pattern for the DynamicAccessList, using a string of bytes.	

]

8.13.1.1.3.2 MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::Byte)**[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	Returns an instance of a diagnostic service pattern builder.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This function is used by the AA to construct a pattern for the DynamicAccessList, using only the SID.	

]

8.13.1.1.3.3 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.	

]

8.13.1.1.3.4 operator=(DiagnosticServiceDynamicAccessList&&) &**[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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>auto operator= (DiagnosticServiceDynamicAccessList &&other) & noexcept -> DiagnosticServiceDynamicAccessList &;</code>	
Parameters (in):	other	Object to move-assign from.
Return value:	DiagnosticServiceDynamicAccessList	DiagnosticServiceDynamicAccessList& Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of DiagnosticServiceDynamicAccessList.	

]

8.13.1.1.3.5 operator=(DiagnosticServiceDynamicAccessList const&) &**[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:	<code>class ara::diag::DiagnosticServiceDynamicAccessList</code>	
Syntax:	<code>auto operator= (DiagnosticServiceDynamicAccessList const &other) & noexcept -> DiagnosticServiceDynamicAccessList &;</code>	
Parameters (in):	other	Object to copy-assign from.
Return value:	DiagnosticServiceDynamicAccessList	DiagnosticServiceDynamicAccessList& Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator of DiagnosticServiceDynamicAccessList.	

]

8.14 Header: ara/diag/download.h

8.14.1 Class: DownloadService

[SWS_DM_00784] Definition of API class ara::diag::DownloadService

Upstream requirements: [RS_Diag_04033](#)

Kind:	class
Port Interfaces:	DiagnosticDownloadInterface
Header file:	#include "ara/diag/download.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DownloadService
Syntax:	<code>class DownloadService {...};</code>
Description:	Download service interface.

8.14.1.1 Public Member Functions

8.14.1.1.1 Special Member Functions

8.14.1.1.1.1 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:	<code>DownloadService (DownloadService &&) noexcept=delete;</code>
Description:	Move constructor of DownloadService.

8.14.1.1.1.2 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:	<code>class ara::diag::DownloadService</code>
Syntax:	<code>DownloadService & operator= (DownloadService &&)=delete;</code>
Description:	Move assignment operator of DownloadService.

]

8.14.1.1.1.3 Destructor

[SWS_DM_00788] Definition of API function `ara::diag::DownloadService::~~DownloadService`

Upstream requirements: [RS_AP_00134](#), [RS_Diag_04033](#)

[

Kind:	function
Header file:	#include "ara/diag/download.h"
Scope:	<code>class ara::diag::DownloadService</code>
Syntax:	<code>virtual ~DownloadService () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class DownloadService.

]

8.14.1.1.2 Constructors

8.14.1.1.2.1 DownloadService(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_00787] Definition of API function ara::diag::DownloadService::DownloadService

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04033](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	<code>class ara::diag::DownloadService</code>	
Syntax:	<code>explicit DownloadService (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticDownloadInterface
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Class for an DownloadService.	

8.14.1.1.2.2 DownloadService(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:	<code>class ara::diag::DownloadService</code>	
Syntax:	<code>DownloadService (DownloadService &)=delete;</code>	





Description:	DownloadService shall be a single not copy-able instance.

]

8.14.1.1.3 Member Functions

8.14.1.1.3.1 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](#)

[

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	<code>class ara::diag::DownloadService</code>	
Syntax:	<pre>virtual ara::core::Future< OperationOutput > DownloadData (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	transferRequestParameterRecord	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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< OperationOutput ></code>	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 <code>OperationOutput.responseData</code> will be placed after the blockSequenceCounter as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for TransferData following a previous RequestDownload.	

]

8.14.1.1.3.2 Offer

[SWS_DM_00792] Definition of API function ara::diag::DownloadService::OfferUpstream requirements: [RS_AP_00119](#), [RS_AP_00139](#), [RS_Diag_04033](#)

[

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	<code>class ara::diag::DownloadService</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.14.1.1.3.3 RequestDownload

[SWS_DM_00789] Definition of API function ara::diag::DownloadService::RequestDownloadUpstream requirements: [RS_AP_00119](#), [RS_AP_00138](#), [RS_Diag_04033](#), [RS_Diag_04170](#)

[

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	<code>class ara::diag::DownloadService</code>	
Syntax:	<code>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;</code>	
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	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	A Future, which either gets readied to void (for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for RequestDownload.	

]

8.14.1.1.3.4 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](#)

[

Kind:	function	
Header file:	#include "ara/diag/download.h"	
Scope:	<code>class ara::diag::DownloadService</code>	
Syntax:	<pre>virtual ara::core::Future< OperationOutput > RequestDownloadExit (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	transferRequestParameterRecord	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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestTransferExit.	

]

8.14.1.1.3.5 StopOffer

[SWS_DM_00793] Definition of API function ara::diag::DownloadService::StopOffer

Upstream requirements: [RS_Diag_04033](#)

[

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.

]

8.14.1.1.3.6 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.

]

8.14.2 Struct: OperationOutput

[SWS_DM_00785] Definition of API class ara::diag::DownloadService::OperationOutput

Upstream requirements: [RS_Diag_04033](#)

[

Kind:	struct
Header file:	#include "ara/diag/download.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	<code>class ara::diag::DownloadService</code>
Symbol:	OperationOutput
Syntax:	<code>struct OperationOutput {...};</code>
Description:	Response data of positive response message.

]

8.14.2.1 Public Member Variables

8.14.2.1.1 responseData

[SWS_DM_00786] Definition of API variable ara::diag::DownloadService::OperationOutput::responseData

Upstream requirements: [RS_Diag_04033](#)

[

Kind:	variable
Header file:	#include "ara/diag/download.h"
Scope:	<code>struct ara::diag::DownloadService::OperationOutput</code>
Symbol:	responseData
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Description:	Content of positive response 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.

]

8.15 Header: ara/diag/dynamic_access_list_diag_service_builder.h

8.15.1 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:	<code>class DynamicAccessListDiagServiceBuilder final {...};</code>
Description:	Definition of the DynamicAccessListDiagServiceBuilder class, which is used by the application to build a DynamicAccessList.

]

8.15.1.1 Public Member Types

8.15.1.1.1 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
Symbol:	Byte
Syntax:	<code>using Byte = std::uint8_t;</code>
Description:	Type alias of a single diagnostic service pattern element in the DynamicAccessList.

]

8.15.1.1.2 Type Alias: ByteString

[SWS_DM_01168] Definition of API type `ara::diag::DynamicAccessListDiagServiceBuilder::ByteString`

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	type alias
Header file:	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
Scope:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
Symbol:	ByteString
Syntax:	<code>using ByteString = ara::core::Span<Byte>;</code>
Description:	Type alias of a sequence of diagnostic service pattern elements in the DynamicAccessList.

]

8.15.1.2 Public Member Functions

8.15.1.2.1 Special Member Functions

8.15.1.2.1.1 Move Constructor

[SWS_DM_01170] Definition of API function `ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder`

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
Scope:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
Syntax:	<code>DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagServiceBuilder &&other) noexcept;</code>
Parameters (in):	other Object to move-construct from.
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Move constructor of DynamicAccessListDiagServiceBuilder.

]

8.15.1.2.1.2 Destructor

[SWS_DM_01174] Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::~DynamicAccessListDiagServiceBuilderUpstream 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.

]

8.15.1.2.2 Constructors

**8.15.1.2.2.1 DynamicAccessListDiagServiceBuilder(V,
ara::core::Vector<std::uint8_t>&)****[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.	

]

8.15.1.2.2.2 DynamicAccessListDiagServiceBuilder(DynamicAccessListDiagServiceBuilder const&)**[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 (DynamicAccessListDiagServiceBuilder const &other)=delete;
Description:	Copy constructor of DynamicAccessListDiagServiceBuilder cannot be used.

]

8.15.1.2.3 Member Functions**8.15.1.2.3.1 Add(ByteRange)****[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	Returns 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.	

]

8.15.1.2.3.2 Add(Byte)

[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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>	
Syntax:	<code>auto Add (Byte value) noexcept -> DynamicAccessListDiagServiceBuilder &;</code>	
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	Returns 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.	

]

8.15.1.2.3.3 Add(ByteString)

[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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>	
Syntax:	<code>auto Add (ByteString values) noexcept -> DynamicAccessListDiagServiceBuilder &;</code>	
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:	DynamicAccessListDiagServiceBuilder	Returns 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.	

]

8.15.1.2.3.4 Any

[SWS_DM_01178] Definition of API function `ara::diag::DynamicAccessListDiagServiceBuilder::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 -> DynamicAccessListDiagServiceBuilder &;	
Parameters (in):	numberOfBytesToIgnore	The number of bytes to ignore
Return value:	DynamicAccessListDiagServiceBuilder	Returns 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.	

]

8.15.1.2.3.5 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.	

]

8.15.1.2.3.6 EndsWith(ByteRange)**[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.	

]

8.15.1.2.3.7 EndsWith(Byte)**[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.	

]

8.15.1.2.3.8 operator=(DynamicAccessListDiagServiceBuilder&&)**[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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
Syntax:	<code>auto operator= (DynamicAccessListDiagServiceBuilder &&other) -> DynamicAccessListDiagServiceBuilder &=delete;</code>
Description:	Move assignment operator of DynamicAccessListDiagServiceBuilder.

]

8.15.1.2.3.9 operator=(DynamicAccessListDiagServiceBuilder const&)**[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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
Syntax:	<code>auto operator= (DynamicAccessListDiagServiceBuilder const &other) -> DynamicAccessListDiagServiceBuilder &=delete;</code>
Description:	Copy assignment operator of DynamicAccessListDiagServiceBuilder cannot be used.

]

8.15.2 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>





Symbol:	ByteRange
Syntax:	<code>class ByteRange final {...};</code>
Description:	Represents a single byte value closed range. Type alias of a single diagnostic service pattern element in the DynamicAccessList

]

8.15.2.1 Public Member Functions

8.15.2.1.1 Special Member Functions

8.15.2.1.1.1 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>ByteRange (ByteRange &&other) noexcept;</code>	
Parameters (in):	other	Object to move-assign from.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructor of ByteRange.	

]

8.15.2.1.1.2 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>~ByteRange () noexcept;</code>	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	





Description:	Destructor of ByteRange.

]

8.15.2.1.2 Constructors

8.15.2.1.2.1 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>ByteRange (Byte min, Byte max) noexcept;</code>	
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.	

]

8.15.2.1.2.2 ByteRange

[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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>ByteRange (ByteRange const &other) noexcept;</code>	
Parameters (in):	other	Object to construct-assign from.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Copy constructor of ByteRange.	

]

8.15.2.1.3 Member Functions

8.15.2.1.3.1 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>auto GetMax () const noexcept -> Byte;</code>	
Return value:	Byte	Returns the the highest value.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reports the highest value to match.	

8.15.2.1.3.2 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:	<code>class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange</code>	
Syntax:	<code>auto GetMin () const noexcept -> Byte;</code>	
Return value:	Byte	Returns the lowest value.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reports the lowest value to match.	

8.15.2.1.3.3 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 &;	
Parameters (in):	other	Object to copy-assign from.
Return value:	ByteRange	ByteRange& Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator of ByteRange.	

]

8.15.2.1.3.4 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 &;	
Parameters (in):	other	Object to move-assign from.
Return value:	ByteRange	ByteRange& Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of ByteRange.	

]

8.16 Header: ara/diag/ecu_reset_request.h

8.16.1 Non-Member Types

8.16.1.1 Type Alias: ResetRequestType

[SWS_DM_01007] Definition of API type ara::diag::ResetRequestType

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

[

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.

]

8.16.2 Global Variables

8.16.2.1 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

]

8.16.2.2 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.

]

8.16.2.3 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.

]

8.16.2.4 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:	<code>constexpr ResetRequestType kSoftReset = 0;</code>
Description:	SoftReset.

]

8.16.3 Class: EcuResetRequest

[SWS_DM_01009] Definition of API class `ara::diag::EcuResetRequest`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

[

Kind:	class
Port Interfaces:	DiagnosticEcuResetInterface
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>EcuResetRequest</code>
Syntax:	<code>class EcuResetRequest { ... };</code>
Description:	Service EcuReset Request interface.

]

8.16.3.1 Public Member Functions

8.16.3.1.1 Special Member Functions

8.16.3.1.1.1 Move Constructor

[SWS_DM_01622] Definition of API function `ara::diag::EcuResetRequest::EcuResetRequest`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>
Scope:	<code>class ara::diag::EcuResetRequest</code>
Syntax:	<code>EcuResetRequest (EcuResetRequest &&) noexcept=delete;</code>
Description:	Move constructor of EcuResetRequest.

]

8.16.3.1.1.2 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:	<code>class ara::diag::EcuResetRequest</code>
Syntax:	<code>EcuResetRequest & operator= (EcuResetRequest &&)=delete;</code>
Description:	Move assignment operator of EcuResetRequest.

]

8.16.3.1.1.3 Destructor

[SWS_DM_01011] Definition of API function `ara::diag::EcuResetRequest::~~EcuResetRequest`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

[

Kind:	function
Header file:	#include "ara/diag/ecu_reset_request.h"
Scope:	<code>class ara::diag::EcuResetRequest</code>
Syntax:	<code>virtual ~EcuResetRequest () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of EcuResetRequest.

]

8.16.3.1.2 Constructors

8.16.3.1.2.1 EcuResetRequest(const ara::core::InstanceSpecifier&)

[SWS_DM_01010] Definition of API function ara::diag::EcuResetRequest::EcuResetRequest

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

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	InstanceSpecifier to a PortPrototype of a DiagnosticEcuResetInterface .
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.
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticEcuResetInterface needs to be referenced by a DiagnosticServiceGenericMapping .
	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.
	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.
Description:	Constructor of EcuResetRequest.	

8.16.3.1.2.2 EcuResetRequest(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.	

8.16.3.1.3 Member Functions

8.16.3.1.3.1 EnableRapidShutdown

[SWS_DM_01012] Definition of API function `ara::diag::EcuResetRequest::EnableRapidShutdown`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	<code>class ara::diag::EcuResetRequest</code>	
Syntax:	<pre>virtual ara::core::Future< void > EnableRapidShutdown (bool enable, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Interface for subFunction En-/DisableRapidShutdown.	

8.16.3.1.3.2 ExecuteReset

[SWS_DM_01014] Definition of API function `ara::diag::EcuResetRequest::ExecuteReset`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	<code>class ara::diag::EcuResetRequest</code>	
Syntax:	<pre>virtual ara::core::Future< void > ExecuteReset (const MetaInfo &meta Info) noexcept=0;</pre>	
Parameters (in):	metaInfo	MetaInfo of the request.





Return value:	ara::core::Future< void >	Return nothing or an error. In case the parameter DiagnosticEcuResetClass.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:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	process which is instantiating this interface, has to execute the requested reset.	

]

8.16.3.1.3.3 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.16.3.1.3.4 RequestReset

[SWS_DM_01013] Definition of API function ara::diag::EcuResetRequest::RequestReset

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

[

Kind:	function	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	virtual ara::core::Future< void > RequestReset (ResetRequestType reset Type, ara::core::Optional< std::uint8_t > id, const MetaInfo &meta Info, 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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for any EcuReset subFunction, except En-/DisableRapidShutdown. The process, which is instantiating this interface, needs to evaluate 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 <code>ExecuteReset()</code> trigger.	

8.16.3.1.3.5 StopOffer

[SWS_DM_01017] Definition of API function `ara::diag::EcuResetRequest::StopOffer`

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04279](#)

Kind:	function
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>
Scope:	<code>class ara::diag::EcuResetRequest</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

8.16.3.1.3.6 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.

]

8.17 Header: ara/diag/external_authentication.h**8.17.1 Class: ExternalAuthentication****[SWS_DM_01191] Definition of API class ara::diag::ExternalAuthentication***Upstream requirements:* [RS_Diag_04251](#)

[

Kind:	class
Port Interfaces:	DiagnosticExternalAuthenticationInterface
Header file:	#include "ara/diag/external_authentication.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	ExternalAuthentication
Syntax:	<code>class ExternalAuthentication final {...};</code>
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.

]

8.17.1.1 Public Member Types

8.17.1.1.1 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:	<code>class ara::diag::ExternalAuthentication</code>
Symbol:	Address
Syntax:	<code>using Address = std::uint16_t;</code>
Description:	Alias for tester address.

]

8.17.1.2 Public Member Functions

8.17.1.2.1 Special Member Functions

8.17.1.2.1.1 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:	<code>class ara::diag::ExternalAuthentication</code>
Syntax:	<code>ExternalAuthentication (ExternalAuthentication &&other) noexcept=default;</code>
Parameters (in):	other Object to move-construct from
Exception Safety:	exception safe
Description:	Move constructor of ExternalAuthentication.

]

8.17.1.2.1.2 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:	<code>class ara::diag::ExternalAuthentication</code>
Syntax:	<code>~ExternalAuthentication () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticServiceDynamicAccessList.

]

8.17.1.2.2 Constructors

8.17.1.2.2.1 ExternalAuthentication(ara::core::InstanceSpecifier)

[SWS_DM_01193] Definition of API function `ara::diag::ExternalAuthentication::ExternalAuthentication`Upstream requirements: [RS_Diag_04251](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/external_authentication.h"	
Scope:	<code>class ara::diag::ExternalAuthentication</code>	
Syntax:	<code>explicit ExternalAuthentication (ara::core::InstanceSpecifier instanceSpecifier) noexcept;</code>	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a DiagnosticExternalAuthenticationInterface 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.
	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.





	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.
Description:	Constructs the port for authentication of diagnostic clients.	

8.17.1.2.2 ExternalAuthentication(ExternalAuthentication const&)

[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.

8.17.1.2.3 Member Functions

8.17.1.2.3.1 Get(Address)

[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< Client Authentication >	Returns the associated diagnostic client authentication object or error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.17.1.2.3.2 Get(const MetaInfo&)

[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 &metaInfo) noexcept;	
Parameters (in):	metaInfo	MetaInfo of the request.
Return value:	ara::core::Result< ClientAuthentication >	Returns the diagnostic client associated authentication object or error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	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.	

]

8.17.1.2.3.3 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 >	Returns 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.	

]

8.17.1.2.3.4 operator=(ExternalAuthentication const&)

[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.

]

8.17.1.2.3.5 operator=(ExternalAuthentication&&) &

[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	ExternalAuthentication& Reference to self.
Exception Safety:	exception safe	
Description:	Move assignment operator of ExternalAuthentication.	

]

8.18 Header: ara/diag/file_transfer.h

8.18.1 Class: FileTransferService

[SWS_DM_01320] Definition of API class ara::diag::FileTransferService

Upstream requirements: [RS_Diag_04135](#)

[

Kind:	class
Port Interfaces:	DiagnosticRequestFileTransferInterface
Header file:	#include "ara/diag/file_transfer.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	FileTransferService
Syntax:	<code>class FileTransferService {...};</code>
Description:	File Transfer service interface.

]

8.18.1.1 Public Member Types

8.18.1.1.1 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:	<code>enum class WriteFileMode : ara::core::Byte {...};</code>	
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.	

]

8.18.1.2 Public Member Functions

8.18.1.2.1 Special Member Functions

8.18.1.2.1.1 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:	<code>class ara::diag::FileTransferService</code>
Syntax:	<code>virtual ~FileTransferService () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class FileTransferService.

]

8.18.1.2.2 Constructors

8.18.1.2.2.1 FileTransferService

[SWS_DM_01325] Definition of API function `ara::diag::FileTransferService::FileTransferService`

Upstream requirements: [RS_Diag_04135](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/file_transfer.h"	
Scope:	<code>class ara::diag::FileTransferService</code>	
Syntax:	<code>FileTransferService (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a DiagnosticRequestFileTransferInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.





	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.
	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.
Description:	Constructor for FileTransferService (inherited)	

]

8.18.1.2.3 Member Functions

8.18.1.2.3.1 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:	<code>class ara::diag::FileTransferService</code>	
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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< void >	A Future with either a void data result to be reported to the UDS client (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestFileTransfer with ModeOfOperation DeleteFile. This method is the complete operation of deleting a file	

]

8.18.1.2.3.2 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:	<code>class ara::diag::FileTransferService</code>	
Syntax:	<code>auto Offer () noexcept -> ara::core::Result< void >;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.18.1.2.3.3 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:	<code>class ara::diag::FileTransferService</code>	
Syntax:	<code>virtual auto RequestReadDirectory (ara::core::String directoryName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< std::tuple< DataTransferReadSession, std::uint64_t > >=0;</code>	
Parameters (in):	directoryName	Path including name of the directory to read
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< std::tuple< DataTransferReadSession, std::uint64_t > ></code>	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 a negative response message).
Exception Safety:	exception safe	





Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestFileTransfer with ModeOfOperation ReadDir. .	

]

8.18.1.2.3.4 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:	<pre>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;</pre>	
Parameters (in):	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< std::tuple< DataTransferReadSession, FileSizes > >	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 a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestFileTransfer with ModeOfOperation ReadFile. 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.	

]

8.18.1.2.3.5 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:	<code>class ara::diag::FileTransferService</code>	
Syntax:	<pre>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;</pre>	
Parameters (in):	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	fileSizes	The compressed/uncompressed files sizes
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< std::tuple< DataTransferWriteSession, std::uint64_t > ></code>	A Future with either an instance of the file writing session and the byte position where to start resuming from to be reported to the UDS client (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for RequestFileTransfer with ModeOfOperation ResumeFile.	

]

8.18.1.2.3.6 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:	<code>class ara::diag::FileTransferService</code>	
Syntax:	virtual auto RequestWriteFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes fileSizes, WriteFile Mode mode, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) 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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< Data TransferWriteSession >	A Future with either an instance of the file writing session to be reported to the UDS client (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestFileTransfer with ModeOfOperation WriteFile.	

]

8.18.1.2.3.7 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.

]

8.18.1.3 Protected Member Functions

8.18.1.3.1 Special Member Functions

8.18.1.3.1.1 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:	thread-safe	
Description:	Move constructs an instance of FileTransferService.	
Visibility:	protected	

]

8.18.1.3.2 Constructors

8.18.1.3.2.1 FileTransferService

[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:	<code>class ara::diag::FileTransferService</code>
Syntax:	<code>FileTransferService (FileTransferService const &)=delete;</code>
Description:	FileTransferService shall be a single not copy-able instance.
Visibility:	protected

]

8.18.1.3.3 Member Functions

8.18.1.3.3.1 operator=(FileTransferService const&)

[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:	<code>class ara::diag::FileTransferService</code>
Syntax:	<code>auto operator= (FileTransferService const &) -> FileTransferService &=delete;</code>
Description:	FileTransferService shall be a single not assignable instance.
Visibility:	protected

]

8.18.1.3.3.2 operator=(FileTransferService&&) &

[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	

]

8.18.2 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.

]

8.18.2.1 Public Member Variables

8.18.2.1.1 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.

]

8.18.2.1.2 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.

]

8.19 Header: ara/diag/generic_data_identifier.h

8.19.1 Class: GenericDataIdentifier

[SWS_DM_00607] Definition of API class ara::diag::GenericDataIdentifier

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	class
Port Interfaces:	DiagnosticDataIdentifierGenericInterface
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:	<code>class GenericDataIdentifier {...};</code>
Description:	Generic DataIdentifier 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_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:	<code>GenericDataIdentifier (GenericDataIdentifier &&) noexcept=delete;</code>
Description:	Move constructor of GenericDataIdentifier.

]

8.19.1.1.1.2 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:	<code>class ara::diag::GenericDataIdentifier</code>
Syntax:	<code>GenericDataIdentifier & operator= (GenericDataIdentifier &&) = delete;</code>
Description:	Move assignment operator of GenericDataIdentifier.

]

8.19.1.1.1.3 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:	<code>class ara::diag::GenericDataIdentifier</code>
Syntax:	<code>virtual ~GenericDataIdentifier () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class GenericDataIdentifier .

]

8.19.1.1.2 Constructors

8.19.1.1.2.1 GenericDataIdentifier(const ara::core::InstanceSpecifier&, DataIdentifierConcurrencyType)

[SWS_DM_00634] Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04169](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/generic_data_identifier.h"	
Scope:	<code>class ara::diag::GenericDataIdentifier</code>	
Syntax:	<code>explicit GenericDataIdentifier (const ara::core::InstanceSpecifier &specifier, DataIdentifierConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticDataIdentifierGenericInterface.
	concurrencyType	Specifies if the interface is implemented as thread safe (<code>kConcurrent</code>) or non-thread safe (<code>kNotConcurrent</code>)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	<code>InstanceSpecifierMappingIntegrityViolation</code>	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.
	<code>PortInterfaceMappingViolation</code>	A PortPrototype that is typed by a DiagnosticDataIdentifierGenericInterface needs to be referenced by a DiagnosticDataPortMapping.
	<code>ProcessMappingViolation</code>	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.
	<code>InstanceSpecifierAlreadyInUseViolation</code>	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process.
Description:	Class for an GenericDataIdentifier.	

8.19.1.1.2.2 GenericDataIdentifier(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:	<code>class ara::diag::GenericDataIdentifier</code>	
Syntax:	<code>GenericDataIdentifier (GenericDataIdentifier &)=delete;</code>	





Description:	GenericDataIdentifier shall be a single not copy-able instance.

└

8.19.1.1.3 Member Functions

8.19.1.1.3.1 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:	<code>class ara::diag::GenericDataIdentifier</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlready Offered</code>	<code>rollback_semantics</code>
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

└

8.19.1.1.3.2 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:	<code>class ara::diag::GenericDataIdentifier</code>	
Syntax:	<code>virtual ara::core::Future< OperationOutput > Read (std::uint16_t data Identifier, const MetaInfo &metaInfo, CancellationHandler cancellation Handler) noexcept=0;</code>	
Parameters (in):	<code>dataIdentifier</code>	the corresponding DataIdentifier
	<code>metaInfo</code>	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future<OperationOutput></code>	A Future with either <code>OperationOutput</code> (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for <code>ReadDataByIdentifier</code> request for this <code>DiagnosticDataIdentifier</code> .	

]

8.19.1.1.3.3 StopOffer

[SWS_DM_00639] Definition of API function `ara::diag::GenericDataIdentifier::StopOffer`

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/generic_data_identifier.h"</code>
Scope:	<code>class ara::diag::GenericDataIdentifier</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This <code>StopOffer</code> will disable the forwarding of request messages from DM.

]

8.19.1.1.3.4 Write

[SWS_DM_00637] Definition of API function ara::diag::GenericDataIdentifier::WriteUpstream 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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< void >	A Future with either void (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for WriteDataByIdentifier request for this DiagnosticDataIdentifier.	

]

8.19.1.1.3.5 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.

└

8.19.2 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:	<code>class ara::diag::GenericDataIdentifier</code>
Symbol:	OperationOutput
Syntax:	<code>struct OperationOutput {...};</code>
Description:	Response data of positive response message.

└

8.19.2.1 Public Member Variables

8.19.2.1.1 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:	<code>struct ara::diag::GenericDataIdentifier::OperationOutput</code>
Symbol:	responseData
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Description:	Content of positive response message (without DataIdentifier)

└

8.20 Header: ara/diag/generic_routine.h

8.20.1 Class: GenericRoutine

[SWS_DM_00605] Definition of API class ara::diag::GenericRoutine

Upstream requirements: [RS_Diag_04169](#), [RS_Diag_04224](#)

[

Kind:	class
Port Interfaces:	DiagnosticRoutineGenericInterface
Header file:	#include "ara/diag/generic_routine.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	GenericRoutine
Syntax:	<code>class GenericRoutine {...};</code>
Description:	Generic Routine 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_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:	<code>GenericRoutine (GenericRoutine &&) noexcept=delete;</code>
Description:	Move constructor of GenericRoutine.

]

8.20.1.1.1.2 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:	<code>class ara::diag::GenericRoutine</code>
Syntax:	<code>GenericRoutine & operator= (GenericRoutine &&)=delete;</code>
Description:	Move assignment operator of GenericRoutine.

]

8.20.1.1.1.3 Destructor

[SWS_DM_00553] Definition of API function `ara::diag::GenericRoutine::~~GenericRoutine`

Upstream requirements: [RS_AP_00134](#), [RS_Diag_04169](#), [RS_Diag_04224](#)

[

Kind:	function
Header file:	#include "ara/diag/generic_routine.h"
Scope:	<code>class ara::diag::GenericRoutine</code>
Syntax:	<code>virtual ~GenericRoutine () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class GenericRoutine .

]

8.20.1.1.2 Constructors

8.20.1.1.2.1 GenericRoutine(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_00552] Definition of API function ara::diag::GenericRoutine::GenericRoutine

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04169](#), [RS_Diag_04224](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	<code>class ara::diag::GenericRoutine</code>	
Syntax:	<code>explicit GenericRoutine (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticRoutineGenericInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Class for an GenericRoutine.	

8.20.1.1.2.2 GenericRoutine(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:	<code>class ara::diag::GenericRoutine</code>	
Syntax:	<code>GenericRoutine (GenericRoutine &)=delete;</code>	





Description:	GenericRoutine shall be a single not copy-able instance.

]

8.20.1.1.3 Member Functions

8.20.1.1.3.1 Offer

[SWS_DM_00557] Definition of API function `ara::diag::GenericRoutine::Offer`

Upstream requirements: [RS_AP_00119](#), [RS_AP_00138](#), [RS_Diag_04169](#), [RS_Diag_04224](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_routine.h"	
Scope:	<code>class ara::diag::GenericRoutine</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlready Offered</code>	rollback_semantics The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.20.1.1.3.2 RequestResults

[SWS_DM_00556] Definition of API function `ara::diag::GenericRoutine::RequestResults`

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:	<code>class ara::diag::GenericRoutine</code>	
Syntax:	<code>virtual ara::core::Future< OperationOutput > RequestResults (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept;</code>	
Parameters (in):	<code>routineId</code>	the corresponding RoutineIdentifier
	<code>requestData</code>	Content of request message (without RoutineIdentifier), Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	<code>metaInfo</code>	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future<OperationOutput></code>	A Future with either <code>OperationOutput</code> (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for RoutineControl with SubFunction RequestResults (0x03) request for this Diagnostic RoutineIdentifier.	

]

8.20.1.1.3.3 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:	<code>class ara::diag::GenericRoutine</code>	
Syntax:	<pre>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;</pre>	
Parameters (in):	<code>routineId</code>	the corresponding RoutineIdentifier
	<code>requestData</code>	Content of request message (without RoutineIdentifier). Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	<code>metaInfo</code>	MetaInfo of the request.
	<code>cancellationHandler</code>	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.





Return value:	ara::core::Future<OperationOutput >	A Future with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RoutineControl with SubFunction Start (0x01) request for this DiagnosticRoutine Identifier.	

]

8.20.1.1.3.4 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	MetaData of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future<OperationOutput >	A Future with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RoutineControl with SubFunction Stop (0x02) request for this DiagnosticRoutine Identifier.	

]

8.20.1.1.3.5 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:	<code>class ara::diag::GenericRoutine</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.20.1.1.3.6 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:	<code>class ara::diag::GenericRoutine</code>
Syntax:	<code>GenericRoutine & operator= (GenericRoutine &)=delete;</code>
Description:	GenericRoutine shall be a single not assignable instance.

]

8.20.2 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:	<code>class ara::diag::GenericRoutine</code>
Symbol:	OperationOutput
Syntax:	<code>struct OperationOutput {...};</code>
Description:	Response data of positive response message.

]

8.20.2.1 Public Member Variables

8.20.2.1.1 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:	<code>#include "ara/diag/generic_routine.h"</code>
Scope:	<code>struct ara::diag::GenericRoutine::OperationOutput</code>
Symbol:	responseData
Type:	<code>ara::core::Vector<std::uint8_t></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Description:	Content of positive response message (without RoutineIdentifier)

]

8.21 Header: `ara/diag/generic_uds_service.h`

8.21.1 Class: GenericUDSService

[SWS_DM_00602] Definition of API class `ara::diag::GenericUDSService`

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	class
Port Interfaces:	DiagnosticGenericUdsInterface
Header file:	<code>#include "ara/diag/generic_uds_service.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace <code>ara::diag</code>
Symbol:	GenericUDSService
Syntax:	<code>class GenericUDSService {...};</code>
Description:	Generic UDS interface.

]

8.21.1.1 Public Member Functions

8.21.1.1.1 Special Member Functions

8.21.1.1.1.1 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:	<code>class ara::diag::GenericUDSService</code>
Syntax:	<code>GenericUDSService (GenericUDSService &&) noexcept=delete;</code>
Description:	Move constructor of GenericUDSService.

]

8.21.1.1.1.2 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:	<code>class ara::diag::GenericUDSService</code>
Syntax:	<code>GenericUDSService & operator= (GenericUDSService &&)=delete;</code>
Description:	Move assignment operator of GenericUDSService.

]

8.21.1.1.1.3 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:	<code>virtual ~GenericUDSService () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of GenericUDSService.

]

8.21.1.1.2 Constructors

8.21.1.1.2.1 `GenericUDSService(const ara::core::InstanceSpecifier&, ConcurrencyType)`

[SWS_DM_00616] Definition of API function `ara::diag::GenericUDSService::GenericUDSService`

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04169](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_uds_service.h"	
Scope:	class ara::diag::GenericUDSService	
Syntax:	<code>explicit GenericUDSService (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticGenericUdsInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.

▽



	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.
Description:	Constructor of GenericUDSService.	

]

8.21.1.1.2.2 GenericUDSService(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:	<code>class ara::diag::GenericUDSService</code>
Syntax:	<code>GenericUDSService (GenericUDSService &)=delete;</code>
Description:	GenericUDSService shall be a single not copy-able instance.

]

8.21.1.1.3 Member Functions

8.21.1.1.3.1 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:	<code>class ara::diag::GenericUDSService</code>	
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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future<OperationOutput></code>	A Result with either a <code>OperationOutput</code> or an error
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for any request message.	

]

8.21.1.1.3.2 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:	<code>class ara::diag::GenericUDSService</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlreadyOffered</code>	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.21.1.1.3.3 StopOffer

[SWS_DM_00620] Definition of API function ara::diag::GenericUDSService::StopOffer

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.

]

8.21.1.1.3.4 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.

]

8.21.2 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:	<code>class ara::diag::GenericUDSService</code>
Symbol:	OperationOutput
Syntax:	<code>struct OperationOutput {...};</code>
Description:	Response data of positive response message.

]

8.21.2.1 Public Member Variables

8.21.2.1.1 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:	<code>#include "ara/diag/generic_uds_service.h"</code>
Scope:	<code>struct ara::diag::GenericUDSService::OperationOutput</code>
Symbol:	responseData
Type:	<code>ara::core::Vector<std::uint8_t></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Description:	Content of positive response message (without SID)

]

8.22 Header: `ara/diag/meta_info.h`

8.22.1 Class: MetaInfo

[SWS_DM_00971] Definition of API class `ara::diag::MetaInfo`

Upstream requirements: [RS_Diag_04170](#)

[

Kind:	class
Header file:	<code>#include "ara/diag/meta_info.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace <code>ara::diag</code>
Symbol:	MetaInfo
Syntax:	<code>class MetaInfo final {...};</code>
Description:	MetaInfo interface.

]

8.22.1.1 Public Member Types

8.22.1.1.1 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:	<code>class ara::diag::MetaInfo</code>
Symbol:	Context
Syntax:	<code>using Context = std::uint32_t;</code>
Description:	Definition of possible call context.

]

8.22.1.2 Public Member Variables

8.22.1.2.1 kDiagnosticCommunication

[SWS_DM_01342] Definition of API variable `ara::diag::MetaInfo::kDiagnosticCommunication`

Upstream requirements: [RS_Diag_04170](#)

[

Kind:	variable
Header file:	#include "ara/diag/meta_info.h"
Scope:	<code>class ara::diag::MetaInfo</code>
Symbol:	kDiagnosticCommunication
Type:	MetaInfo::Context
Syntax:	<code>MetaInfo::Context kDiagnosticCommunication = 0x01;</code>
Description:	Service request in DCM context.

]

8.22.1.2.2 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:	<code>class ara::diag::MetaInfo</code>
Symbol:	kDoIP
Type:	MetaInfo::Context
Syntax:	<code>MetaInfo::Context kDoIP = 0x03;</code>
Description:	For reading VIN.

]

8.22.1.2.3 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:	<code>class ara::diag::MetaInfo</code>
Symbol:	kFaultMemory
Type:	MetaInfo::Context
Syntax:	<code>MetaInfo::Context kFaultMemory = 0x02;</code>
Description:	For DIDs in Snapshots.

]

8.22.1.2.4 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:	<code>class ara::diag::MetaInfo</code>
Symbol:	kSovd
Type:	MetaInfo::Context



△

Syntax:	<code>MetaInfo::Context kSovd = 0x04;</code>
Description:	Service request in SOVD context.

]

8.22.1.3 Public Member Functions

8.22.1.3.1 Special Member Functions

8.22.1.3.1.1 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:	<code>#include "ara/diag/meta_info.h"</code>
Scope:	<code>class ara::diag::MetaInfo</code>
Syntax:	<code>MetaInfo (const MetaInfo &)=delete;</code>
Description:	Copy Constructor of MetaInfo cannot be used.

]

8.22.1.3.1.2 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:	<code>#include "ara/diag/meta_info.h"</code>
Scope:	<code>class ara::diag::MetaInfo</code>
Syntax:	<code>MetaInfo () noexcept=delete;</code>
Description:	Constructor of MetaInfo cannot be used.

]

8.22.1.3.1.3 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:	<code>class ara::diag::MetaInfo</code>	
Syntax:	<code>MetaInfo (MetaInfo &&obj) noexcept;</code>	
Parameters (in):	obj	object to be moved
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move Constructor of MetaInfo.	

]

8.22.1.3.1.4 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:	<code>class ara::diag::MetaInfo</code>	
Syntax:	<code>MetaInfo & operator= (const MetaInfo &)=delete;</code>	
Description:	Copy Assignment Operator of MetaInfo cannot be used.	

]

8.22.1.3.1.5 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:	<code>class ara::diag::MetaInfo</code>	
Syntax:	<code>MetaInfo & operator= (MetaInfo &&other) & noexcept;</code>	
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.

]

8.22.1.3.1.6 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:	<code>class ara::diag::MetaInfo</code>
Syntax:	<code>~MetaInfo () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Default destructor.

]

8.22.1.3.2 Member Functions

8.22.1.3.2.1 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:	<code>class ara::diag::MetaInfo</code>
Syntax:	<code>Context GetContext () const noexcept;</code>
Return value:	Context Returns the context.
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Get the context of the invocation.

]

8.22.1.3.2.2 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.	

]

8.23 Header: ara/diag/release_handler.h

8.23.1 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 cancelled.

]

8.23.1.1 Public Member Types

8.23.1.1.1 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:	<code>class ara::diag::ReleaseHandler</code>
Symbol:	ReleaseHandlerSetNotifier
Syntax:	<code>using ReleaseHandlerSetNotifier = std::function<void(void)>;</code>
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the shared resource can be freed.

]

8.23.1.2 Public Member Functions

8.23.1.2.1 Special Member Functions

8.23.1.2.1.1 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:	<code>class ara::diag::ReleaseHandler</code>
Syntax:	<code>ReleaseHandler ()=delete;</code>
Description:	Default constructor of ReleaseHandler cannot be used.

]

8.23.1.2.1.2 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:	<code>class ara::diag::ReleaseHandler</code>	
Syntax:	<code>ReleaseHandler (ReleaseHandler &&other) noexcept;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructs instance of class.	

8.23.1.2.1.3 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:	<code>class ara::diag::ReleaseHandler</code>	
Syntax:	<code>ReleaseHandler & operator= (ReleaseHandler &&other) & noexcept;</code>	
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.	

8.23.1.2.1.4 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:	<code>class ara::diag::ReleaseHandler</code>
Syntax:	<code>~ReleaseHandler () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Default destructor.

]

8.23.1.2.2 Constructors

8.23.1.2.2.1 ReleaseHandler

[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:	<code>class ara::diag::ReleaseHandler</code>
Syntax:	<code>ReleaseHandler (ReleaseHandler const &)=delete;</code>
Description:	Copy constructor of ReleaseHandler cannot be used.

]

8.23.1.2.3 Member Functions

8.23.1.2.3.1 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:	<code>class ara::diag::ReleaseHandler</code>	
Syntax:	<code>bool MayRelease () const noexcept;</code>	
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.	

8.23.1.2.3.2 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:	<code>class ara::diag::ReleaseHandler</code>	
Syntax:	<code>void SetNotifier (ReleaseHandlerSetNotifier notifier) noexcept;</code>	
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.	
Notes:	The passed in function does not throw any exceptions. The passed notifier must contain a callable object.	

8.23.1.2.3.3 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:	<code>class ara::diag::ReleaseHandler</code>
Syntax:	<code>ReleaseHandler & operator= (ReleaseHandler const &)=delete;</code>
Description:	Copy assignment operator of ReleaseHandler cannot be used.

]

8.24 Header: ara/diag/security_access.h

8.24.1 Non-Member Types

8.24.1.1 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.	

]

8.24.2 Class: SecurityAccess

[SWS_DM_00761] Definition of API class ara::diag::SecurityAccess

Upstream requirements: [RS_Diag_04005](#)

[

Kind:	class
Port Interfaces:	DiagnosticSecurityLevelInterface
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.

]

8.24.2.1 Public Member Functions

8.24.2.1.1 Special Member Functions

8.24.2.1.1.1 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.

]

8.24.2.1.1.2 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:	<code>class ara::diag::SecurityAccess</code>
Syntax:	<code>SecurityAccess & operator= (SecurityAccess &&)=delete;</code>
Description:	Move assignment operator of SecurityAccess.

]

8.24.2.1.1.3 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:	<code>class ara::diag::SecurityAccess</code>
Syntax:	<code>virtual ~SecurityAccess () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SecurityAccess .

]

8.24.2.1.2 Constructors

8.24.2.1.2.1 SecurityAccess(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_00762] Definition of API function ara::diag::SecurityAccess::SecurityAccess

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04005](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/security_access.h"	
Scope:	<code>class ara::diag::SecurityAccess</code>	
Syntax:	<code>explicit SecurityAccess (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticSecurityLevelInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Constructor of SecurityAccess.	

8.24.2.1.2.2 SecurityAccess(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:	<code>class ara::diag::SecurityAccess</code>	
Syntax:	<code>SecurityAccess (SecurityAccess &)=delete;</code>	





Description:	SecurityAccess shall be a single not copy-able instance.

]

8.24.2.1.3 Member Functions

8.24.2.1.3.1 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:	<code>class ara::diag::SecurityAccess</code>	
Syntax:	<pre>virtual ara::core::Future< KeyCompareResultType > CompareKey (ara::core::Span< const std::uint8_t > key, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< Key CompareResultType ></code>	Result of the key validation.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	This method is called, when a diagnostic request has been finished, to notify about the outcome.	

]

8.24.2.1.3.2 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:	<code>class ara::diag::SecurityAccess</code>	
Syntax:	<pre>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;</pre>	
Parameters (in):	securityAccessDataRecord	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 is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. 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 non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for any request message.	

8.24.2.1.3.3 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:	<code>class ara::diag::SecurityAccess</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
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
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.24.2.1.3.4 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:	<code>class ara::diag::SecurityAccess</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.24.2.1.3.5 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:	<code>class ara::diag::SecurityAccess</code>
Syntax:	<code>SecurityAccess & operator= (SecurityAccess &)=delete;</code>
Description:	SecurityAccess shall be a single not assignable instance.

]

8.25 Header: ara/diag/service_validation.h

8.25.1 Non-Member Types

8.25.1.1 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 successful.	
	kResNegNotOk	= 0x03 Negative response has not been sent out successfully.	
	kResPosSuppressed	= 0x04 Positive answer suppressed.	
	kResNegSuppressed	= 0x05 Negative answer suppressed.	
	kCanceled	= 0x06 Processing is cancelled.	
	kNoProcessingNoResponse	= 0x07 Processing rejected in Validation.	
	Description:	Represents the status of the service processing.	

8.25.2 Class: ServiceValidation

[SWS_DM_00771] Definition of API class ara::diag::ServiceValidation

Upstream requirements: [RS_Diag_04199](#)

[

Kind:	class
Port Interfaces:	DiagnosticServiceValidationInterface
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.

]

8.25.2.1 Public Member Functions

8.25.2.1.1 Special Member Functions

8.25.2.1.1.1 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.

]

8.25.2.1.1.2 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:	<code>class ara::diag::ServiceValidation</code>
Syntax:	<code>ServiceValidation & operator= (ServiceValidation &&)=delete;</code>
Description:	Move assignment operator of ServiceValidation.

]

8.25.2.1.1.3 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:	<code>class ara::diag::ServiceValidation</code>
Syntax:	<code>virtual ~ServiceValidation () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of ServiceValidation .

]

8.25.2.1.2 Constructors

8.25.2.1.2.1 ServiceValidation(const ara::core::InstanceSpecifier&)

[SWS_DM_00772] Definition of API function ara::diag::ServiceValidation::ServiceValidation

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04199](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

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 a PortPrototype of a 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.
	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.
	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.
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.	

8.25.2.1.2.2 ServiceValidation(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.	

8.25.2.1.3 Member Functions

8.25.2.1.3.1 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:	<code>class ara::diag::ServiceValidation</code>	
Syntax:	virtual void Confirmation (<code>ConfirmationStatusType</code> status, const <code>MetaInfo</code> &metaInfo) 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.	

8.25.2.1.3.2 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:	<code>class ara::diag::ServiceValidation</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlreadyOffered</code>	rollback_semantics The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

8.25.2.1.3.3 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.

]

8.25.2.1.3.4 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:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for any request message.	

]

8.25.2.1.3.5 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:	<code>class ara::diag::ServiceValidation</code>
Syntax:	<code>ServiceValidation & operator= (ServiceValidation &)=delete;</code>
Description:	ServiceValidation shall be a single not assignable instance.

]

8.26 Header: ara/diag/transmit_certificate.h

8.26.1 Class: TransmitCertificate

[SWS_DM_01961] Definition of API class ara::diag::TransmitCertificate

Upstream requirements: [RS_Diag_04251](#)

[

Kind:	class
Port Interfaces:	DiagnosticTransmitCertificateInterface
Header file:	#include "ara/diag/transmit_certificate.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	TransmitCertificate
Syntax:	<code>class TransmitCertificate {...};</code>
Description:	Class to implement the subfunction TransmitCertificate (29 04) of UDS service Authentication as interface to application. This class implements a so called generic interface, where the application has means to handle one or more certificateEvaluationId transferred certificates.

]

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_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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>TransmitCertificate (TransmitCertificate &&) noexcept=delete;</code>
Description:	Move constructor of TransmitCertificate.

]

8.26.1.1.1.2 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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>TransmitCertificate & operator= (TransmitCertificate &&)=delete;</code>
Description:	Move assignment operator of TransmitCertificate.

]

8.26.1.1.1.3 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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>virtual ~TransmitCertificate () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructor of TransmitCertificate.

]

8.26.1.1.2 Constructors

8.26.1.1.2.1 `TransmitCertificate(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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>TransmitCertificate (TransmitCertificate &)=delete;</code>
Description:	Authentication shall be a single not copy-able instance.

]

8.26.1.1.2.2 TransmitCertificate(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_01962] Definition of API function ara::diag::TransmitCertificate::TransmitCertificate

Upstream requirements: [RS_Diag_04251](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

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 DiagnosticTransmitCertificateInterface service instance in the manifest.
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Constructor of TransmitCertificate.	

8.26.1.1.3 Member Functions

8.26.1.1.3.1 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 >	Returns nothing or an error.





Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.26.1.1.3.2 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:	<code>class ara::diag::TransmitCertificate</code>	
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 evaluation 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 is used to - Query the current cancellation status by calling cancellationHandler.IsCanceled. Returns true in case, the current conversation has been cancelled and false otherwise. - Register a notifier function by calling cancellationHandler.SetNotifier. 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	This method is used to process certificates send by a diagnostic tester, via the UDS subfunction 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.	

]

8.26.1.1.3.3 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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.26.1.1.3.4 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:	<code>class ara::diag::TransmitCertificate</code>
Syntax:	<code>TransmitCertificate & operator= (TransmitCertificate &)=delete;</code>
Description:	Authentication shall be a single not assignable instance.

]

8.27 Header: ara/diag/upload.h

8.27.1 Class: UploadService

[SWS_DM_00794] Definition of API class ara::diag::UploadService

Upstream requirements: [RS_Diag_04033](#)

[

Kind:	class
Port Interfaces:	DiagnosticUploadInterface
Header file:	#include "ara/diag/upload.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	UploadService
Syntax:	<code>class UploadService {...};</code>
Description:	Upload service interface.

]

8.27.1.1 Public Member Functions

8.27.1.1.1 Special Member Functions

8.27.1.1.1.1 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:	<code>UploadService (UploadService &&) noexcept=delete;</code>
Description:	Move constructor of UploadService.

]

8.27.1.1.1.2 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:	<code>class ara::diag::UploadService</code>
Syntax:	<code>UploadService & operator= (UploadService &&)=delete;</code>
Description:	Move assignment operator of UploadService.

]

8.27.1.1.1.3 Destructor

[SWS_DM_00798] Definition of API function ara::diag::UploadService::~~UploadService

Upstream requirements: [RS_AP_00134](#), [RS_Diag_04033](#)

[

Kind:	function
Header file:	#include "ara/diag/upload.h"
Scope:	<code>class ara::diag::UploadService</code>
Syntax:	<code>virtual ~UploadService () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class UploadService.

]

8.27.1.1.2 Constructors

8.27.1.1.2.1 UploadService(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:	<code>class ara::diag::UploadService</code>





Syntax:	UploadService (UploadService &)=delete;
Description:	UploadService shall be a single not copy-able instance.

]

8.27.1.1.2.2 UploadService(const ara::core::InstanceSpecifier&, Concurrency Type)

[SWS_DM_00797] Definition of API function ara::diag::UploadService::Upload Service

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04033](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

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 a PortPrototype of a DiagnosticUploadInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Class for an UploadService.	

]

8.27.1.1.3 Member Functions

8.27.1.1.3.1 Offer

[SWS_DM_00802] Definition of API function `ara::diag::UploadService::Offer`

Upstream requirements: [RS_AP_00119](#), [RS_AP_00139](#), [RS_Diag_04033](#)

[

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	<code>class ara::diag::UploadService</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlready Offered</code>	<code>rollback_semantics</code>
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.27.1.1.3.2 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](#)

[

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	<code>class ara::diag::UploadService</code>	
Syntax:	<code>virtual ara::core::Future< void > RequestUpload (std::uint8_t data FormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const Meta Info &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</code>	
Parameters (in):	<code>dataFormatIdentifier</code>	UDS dataFormat Identifier
	<code>addressAndLength FormatIdentifier</code>	UDS addressAndLengthFormatIdentifier
	<code>memoryAddressAndSize</code>	memoryAddress and memorySize part of the request. Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
	<code>metaInfo</code>	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	A Result with either void (for a positive response message) or an UDS NRC value (for a negative response message).
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for RequestUpload.	

]

8.27.1.1.3.3 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](#)

[

Kind:	function	
Header file:	#include "ara/diag/upload.h"	
Scope:	<code>class ara::diag::UploadService</code>	
Syntax:	<pre>virtual ara::core::Future< OperationOutput > RequestUploadExit (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	transferRequestParameterRecord	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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using DiagUdsNrcErrc .	
Description:	Called for RequestTransferExit.	

]

8.27.1.1.3.4 StopOffer

[SWS_DM_00803] Definition of API function ara::diag::UploadService::StopOffer

Upstream requirements: [RS_Diag_04033](#)

[

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.

]

8.27.1.1.3.5 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](#)

[

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	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future<OperationOutput></code>	A Future, which either gets readied to <code>OperationOutput</code> (transfer <code>ResponseParameterRecord</code> for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for a negative response message). Data in <code>OperationOutput.responseData</code> will be placed after block <code>SequenceCounter</code> as <code>transferResponseParameterRecord</code> in the positive response.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Called for <code>TransferData</code> following a previous <code>RequestUpload</code> .	

]

8.27.1.1.3.6 operator=

[SWS_DM_01671] Definition of API function `ara::diag::UploadService::operator=`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/upload.h"</code>
Scope:	<code>class ara::diag::UploadService</code>
Syntax:	<code>UploadService & operator= (UploadService &)=delete;</code>
Description:	UploadService shall be a single not assignable instance.

]

8.27.2 Struct: OperationOutput

[SWS_DM_00795] Definition of API class ara::diag::UploadService::OperationOutput

Upstream requirements: [RS_Diag_04033](#)

[

Kind:	struct
Header file:	#include "ara/diag/upload.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	<code>class ara::diag::UploadService</code>
Symbol:	OperationOutput
Syntax:	<code>struct OperationOutput { ... };</code>
Description:	Response data of positive response message.

]

8.27.2.1 Public Member Variables

8.27.2.1.1 responseData

[SWS_DM_00796] Definition of API variable ara::diag::UploadService::OperationOutput::responseData

Upstream requirements: [RS_Diag_04033](#)

[

Kind:	variable
Header file:	#include "ara/diag/upload.h"
Scope:	<code>struct ara::diag::UploadService::OperationOutput</code>
Symbol:	responseData
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Description:	Content of positive response 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.

]

8.28 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 <code>DiagnosticDataElementInterface</code> , 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 <code>DiagnosticDataElementInterface.shortName</code> converted to lower-case.
Example:	<pre>// 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)</pre>	

]

8.28.1 Namespaces

8.28.1.1 {<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 <code>SymbolProps</code> aggregated in the role <code>PortInterface.namespace</code> . For each <code>namespace</code> in the ordered list: <code>namespace[N+1]</code> shall be an inner namespace of <code>namespace[N]</code> converted to lower-case.	
Descriptors:	{<namespace-list-data-element>}	as per <code>namespace</code> in the ordered list: <code>namespace[N+1]</code> shall be an inner namespace of <code>namespace[N]</code> converted to lower-case.

]

8.28.2 Class: {<data-element-interface-name-x>}**[SWS_DM_00603] Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name-x>}***Status:* DRAFT*Upstream requirements:* RS_Diag_04169

[

Kind:	class		
Port Interfaces:	DiagnosticDataElementInterface		
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"		
Forwarding header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}/{<data-element-shortname-lower>}_fwd.h"		
Scope:	namespace {<namespace-list-data-element>}		
Symbol:	{<data-element-interface-name-x>}		
Syntax:	class {<data-element-interface-name-x>} {...};		
Description:	DiagnosticDataElementInterface class		
Descriptors:	<table border="1"> <tr> <td>{<data-element-interface-name-x>}</td><td>The DiagnosticDataElementInterface.shortName</td></tr> </table>	{<data-element-interface-name-x>}	The DiagnosticDataElementInterface.shortName
{<data-element-interface-name-x>}	The DiagnosticDataElementInterface.shortName		

]

8.28.2.1 Public Member Variables**8.28.2.1.1 {<data-element-out-arg-symbol>}****[SWS_DM_80023] Definition of API variable {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<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-x>}				
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 <i>out</i> argument in an Output .				
Descriptors:	<table border="1"> <tr> <td>{<data-element-out-arg-type>}</td><td>The ClientServerOperation.argument.type, mapped to a C++ data type according to [27].</td></tr> <tr> <td>{<data-element-out-arg-symbol>}</td><td>Symbol name of the struct element as given by ClientServerOperation.ArgumentDataPrototype.shortName</td></tr> </table>	{<data-element-out-arg-type>}	The ClientServerOperation.argument.type , mapped to a C++ data type according to [27].	{<data-element-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation.ArgumentDataPrototype.shortName
{<data-element-out-arg-type>}	The ClientServerOperation.argument.type , mapped to a C++ data type according to [27].				
{<data-element-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 Member Functions

8.28.2.2.1.1 Offer

[SWS_DM_00597] Definition of API function `{<namespace-list-data-element>}::<data-element-interface-name-x>::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-x>	
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:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

8.28.2.2.1.2 Read

[SWS_DM_00596] Definition of API function `{<namespace-list-data-element>}::<data-element-interface-name-x>::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-x>	
Syntax:	virtual ara::core::Future< {<data-element-name>}ReadOutput > Read (const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept=0;	
Parameters (in):	metaInfo	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< {
<data-element-name>} ReadOutput ></code>	- If successful: an <code>ara::core::Future</code> containing a <code>ReadOutput</code> object as per [SWS_DM_00581] - If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>DiagUdsNrcErrc</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Provision of a Read DataElement (<code>DiagnosticDataElementInterface</code> in the role <code>read</code>)	
Descriptors:	<code>{<data-element-name>}</code>	Name of a DataElement created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticDataElementInterface</code> in the role <code>read</code> .
Example:	<pre>// Example: virtual ara::core::Future<SomeDataElementReadOutput> Read(const ara::diag::MetaInfo& metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept = 0;</pre>	

8.28.2.2.1.3 StopOffer

[SWS_DM_00617] Definition of API function `{<namespace-list-data-element>}::{<data-element-interface-name-x>}::StopOffer`

Upstream requirements: RS_Diag_04169

Kind:	function
Header file:	<code>#include "<data-element-directory-path>/<data-element-shortname-lower>.h"</code>
Scope:	<code>class {<namespace-list-data-element>}::{<data-element-interface-name-x>}</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM

8.28.2.2.1.4 operator=

[SWS_DM_01659] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::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-x>}	
Syntax:	{<data-element-interface-name-3>} & operator= (const {<data-element-interface-name-3>} &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	{<data-element-interface-name-3>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

]

8.28.2.2.1.5 operator=

[SWS_DM_01660] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::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-x>}	
Syntax:	{<data-element-interface-name-5>} & operator= ({<data-element-interface-name-5>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<data-element-interface-name-5>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

]

8.28.2.2.1.6 {<data-element-interface-name-1>}

[SWS_DM_00587] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<data-element-interface-name-1>}

Upstream requirements: RS_AP_00137, RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173

Kind:	function	
Header file:	#include "{<data-element-directory-path>}/{<data-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name-x>}	
Syntax:	{<data-element-interface-name-1>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Construct a DataElement from an ara::core::InstanceSpecifier	
Descriptors:	{<data-element-interface-name-1>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

8.28.2.2.1.7 {<data-element-interface-name-2>}

[SWS_DM_01661] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<data-element-interface-name-2>}

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-x>}	
Syntax:	{<data-element-interface-name-2>} (const <data-element-interface-name-2> &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	{<data-element- interface-name-2>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

]

8.28.2.2.1.8 {<data-element-interface-name-4>}

[SWS_DM_01662] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<data-element-interface-name-4>}

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-x>}	
Syntax:	{<data-element-interface-name-4>} ({<data-element-interface-name-4> &&other)=delete;	
Description:	Move constructor	
Descriptors:	{<data-element- interface-name-4>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

]

8.28.2.2.1.9 ~{<data-element-interface-name-6>}

[SWS_DM_00588] Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::~~{<data-element-interface-name-6>}

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-x>}	





Syntax:	virtual ~{<data-element-interface-name-6>} () noexcept;	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Destruction of a <code>DataElement</code>	
Descriptors:	{<data-element-interface-name-6>}	As per {<data-element-interface-name-x>} in [SWS_DM_00603]

]

8.28.3 Struct: {<data-element-name>}ReadOutput

[SWS_DM_00580] Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<data-element-name>}ReadOutput

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>}/{<data-element-shortname-lower>}_fwd.h"	
Scope:	class {<namespace-list-data-element>}::{<data-element-interface-name-x>}	
Symbol:	{<data-element-name>}ReadOutput	
Syntax:	struct {<data-element-name>}ReadOutput {...};	
Description:	Structure wrapping the <i>out</i> arguments for a <code>ReadOutput</code>	
Descriptors:	{<data-element-name>}	As per {<data-element-name>} in [SWS_DM_00596].
	{<data-element-out-args-members-ordered>}	Shown as "..." in Syntax. Each element is an <i>argument</i> in the ordered list of <i>arguments</i> , with <i>direction</i> == <i>out</i> , where [SWS_DM_80023] applies for each.

]

8.29 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:	<pre>// 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)</pre>	

]

8.29.1 Namespaces

8.29.1.1 {<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 <code>namespace</code> in the ordered list: <code>namespace[N+1]</code> shall be an inner namespace of <code>namespace[N]</code> converted to lower-case.

]

8.29.2 Class: {<data-identifier-interface-name-x>}

[SWS_DM_00601] Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}

Status: DRAFT

Upstream requirements: RS_Diag_04169

[

Kind:	class
Port Interfaces:	<code>DiagnosticDataIdentifierInterface</code>
Header file:	<code>#include "<data-identifier-directory-path>/<data-identifier-shortname-lower>.h"</code>
Forwarding header file:	<code>#include "<data-identifier-directory-path>/<data-identifier-shortname-lower>/<data-identifier-shortname-lower>_fwd.h"</code>
Scope:	namespace {<namespace-list-data-identifier>}
Symbol:	{<data-identifier-interface-name-x>}
Syntax:	<code>class {<data-identifier-interface-name-x>} {...};</code>
Description:	DiagnosticDataIdentifierInterface class
Descriptors:	{<data-identifier-interface-name-x>} The <code>DiagnosticDataIdentifierInterface.shortName</code>

]

8.29.2.1 Public Member Variables

8.29.2.1.1 {<data-identifier-out-arg-symbol>}

[SWS_DM_80013] Definition of API variable {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-out-arg-symbol>}

Status: DRAFT

Upstream requirements: RS_Diag_04224, RS_Diag_04169

[

Kind:	variable
Header file:	<code>#include "<data-identifier-directory-path>/<data-identifier-shortname-lower>.h"</code>
Scope:	<code>class {<namespace-list-data-identifier>}::<data-identifier-interface-name-x></code>





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 <i>out</i> argument in an <i>Output</i> .	
Descriptors:	{<data-identifier-out-arg-type>}	The <code>ClientServerOperation.argument.type</code> , mapped to a C++ data type according to [27].
	{<data-identifier-out-arg-symbol>}	Symbol name of the <code>struct</code> element as given by <code>ClientServerOperation.ArgumentDataPrototype.shortName</code>

]

8.29.2.2 Public Member Functions

8.29.2.2.1 Member Functions

8.29.2.2.1.1 Offer

[SWS_DM_00599] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::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-x>}	
Syntax:	ara::core::Result< void > Offer () noexcept;	
Return value:	ara::core::Result< void >	• If successful: an <code>ara::core::Result</code> containing a <code>ara::core::Result::value_type</code> • If unsuccessful: an <code>ara::core::Result</code> containing an <code>ara::core::Result::error_type</code> i.e. a corresponding <code>ara::diag::DiagOfferErrc</code>
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

]

8.29.2.2.1.2 Read

[SWS_DM_00640] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::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-x>}	
Syntax:	virtual ara::core::Future< {<data-identifier-name>}ReadOutput > Read (const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept=0;	
Parameters (in):	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< {<data-identifier-name>}ReadOutput >	- If successful: an <code>ara::core::Future</code> containing a <code>ReadOutput</code> object as per [SWS_DM_00579] - If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>DiagUdsNrcErrc</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Provision of a reading DataIdentifier (<code>DiagnosticDataIdentifierInterface</code> in the role <code>read</code>).	
Descriptors:	{<data-identifier-name>}	Name of a DataIdentifier created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticDataIdentifierInterface</code> in the role <code>read</code> .
Example:	<pre>// Example: virtual ara::core::Future<SomeDataIdentifierReadOutput> Read(const ara::diag::MetaInfo& metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept = 0;</pre>	

8.29.2.2.1.3 StopOffer

[SWS_DM_00600] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::StopOffer`

Upstream requirements: RS_Diag_04169

[

Kind:	function
Header file:	<code>#include "<data-identifier-directory-path>/{<data-identifier-shortname-lower>}.h"</code>
Scope:	<code>class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM

]

8.29.2.2.1.4 Write

[SWS_DM_00598] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::Write`

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

[

Kind:	function	
Header file:	<code>#include "<data-identifier-directory-path>/{<data-identifier-shortname-lower>}.h"</code>	
Scope:	<code>class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}</code>	
Syntax:	<code>virtual ara::core::Future< void > Write (</code> <code>{<data-identifier-in-arg-derived-type-0toN>}</code> <code>{<data-identifier-in-arg-symbol-0toN>}, const ara::diag::MetaInfo</code> <code>&metaInfo, ara::diag::CancellationHandler cancellationHandler)</code> <code>noexcept=0;</code>	
Parameters (in):	<code>{<data-identifier-in-arg-symbol-0toN>}</code>	The symbol name of method <code>argument[0..N]</code> in the ordered list of method arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.ArgumentDataPrototype[0..N].shortName</code>
	<code>metaInfo</code>	MetaInfo of the request.





	cancellationHandler	This parameter is used to • Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. • Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< void ></code>	• If successful: an <code>ara::core::Future</code> containing a void. • If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>ara::diag::DiagnosticDataIdentifierInterface</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagnosticDataIdentifierInterface</code> .	
Description:	Provision of a writing DataIdentifier (<code>DiagnosticDataIdentifierInterface</code> in the role <code>write</code>).	
Descriptors:	<code>{<derived-type>}</code>	The C++ data type shall be derived according to the following rules: 1. If the (<code>CppImplementationDataType.category == VALUE</code>) or a (<code>CppImplementationDataType.category == TYPE_REFERENCE</code> which transitively type-resolves to a <code>CppImplementationDataType.category == VALUE</code>) and the <code>CppImplementationDataType.shortName</code> is either: • <code>int8_t</code>: as per [SWS_APT_00001] • <code>int16_t</code>: as per [SWS_APT_00004] • <code>int32_t</code>: as per [SWS_APT_00007] • <code>int64_t</code>: as per [SWS_APT_00010] • <code>uint8_t</code>: as per [SWS_APT_00022] • <code>uint16_t</code>: as per [SWS_APT_00025] • <code>uint32_t</code>: as per [SWS_APT_00028] • <code>uint64_t</code>: as per [SWS_APT_00031] • <code>bool</code>: as per [SWS_APT_00049] • <code>float</code>: as per [SWS_APT_00043] • <code>double</code>: as per [SWS_APT_00046] in [28], then <code>{<derived-type>}</code> shall be unchanged, i.e. shall be the mapped C++ data type according to [27]. 2. Otherwise <code>{<derived-type>}</code> shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [29]), to the mapped C++ data type according to [27].
	<code>{<data-identifier-in-arg-derived-type-0toN>}</code>	The data type of method <code>argument[0..N]</code> in the ordered list of method arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.argument[0..N].type</code>, derived as per <code>{<derived-type>}</code>



**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
) noexcept = 0;
```

8.29.2.2.1.5 operator=

[SWS_DM_01664] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::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-x>}	
Syntax:	{<data-identifier-interface-name-5>} & operator= ({<data-identifier-interface-name-5>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<data-identifier-interface-name-5>}	As per {<data-identifier-interface-name-x>} in [SWS_DM_00601]

8.29.2.2.1.6 operator=

[SWS_DM_01663] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::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-x>}	
Syntax:	{<data-identifier-interface-name-3>} & operator= (const {<data-identifier-interface-name-3>} &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	{<data-identifier-interface-name-3>}	As per {<data-identifier-interface-name-x>} in [SWS_DM_00601]

]

8.29.2.2.1.7 {<data-identifier-interface-name-1>}

[SWS_DM_00585] Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-interface-name-1>}

Upstream requirements: RS_AP_00137, RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173

[

Kind:	function	
Header file:	#include "{<data-identifier-directory-path>}/{<data-identifier-shortname-lower>}.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}	
Syntax:	{<data-identifier-interface-name-1>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	Specifies if the interface is implemented as thread safe (<code>kConcurrent</code>) or non-thread safe (<code>kNotConcurrent</code>)..
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.
	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.
	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.
Description:	Construct a DataIdentifier from an <code>ara::core::InstanceSpecifier</code>	
Descriptors:	<code>{<data-identifier-interface-name-1>}</code>	As per <code>{<data-identifier-interface-name-x>}</code> in [SWS_DM_00601]

]

8.29.2.2.1.8 {<data-identifier-interface-name-2>}

[SWS_DM_01665] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-interface-name-2>}`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function	
Header file:	#include "<data-identifier-directory-path>/<data-identifier-shortname-lower>.h"	
Scope:	class <code>{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}</code>	
Syntax:	<code>{<data-identifier-interface-name-2>}</code> (const <code>{<data-identifier-interface-name-2>}</code> &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	<code>{<data-identifier-interface-name-2>}</code>	As per <code>{<data-identifier-interface-name-x>}</code> in [SWS_DM_00601]

]

8.29.2.2.1.9 {<data-identifier-interface-name-4>}

[SWS_DM_01666] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-interface-name-4>}`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function	
Header file:	#include "<data-identifier-directory-path>/<data-identifier-shortname-lower>.h"	
Scope:	class <code>{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}</code>	





Syntax:	<code>{<data-identifier-interface-name-4>} ({<data-identifier-interface-name-4>} &&other)=delete;</code>	
Description:	Move constructor	
Descriptors:	<code>{<data-identifier-interface-name-4>}</code>	As per <code>{<data-identifier-interface-name-x>}</code> in [SWS_DM_00601]

]

8.29.2.2.1.10 ~{<data-identifier-interface-name-6>}

[SWS_DM_00586] Definition of API function `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::~~{<data-identifier-interface-name-6>}`

Status: DRAFT

Upstream requirements: RS_Diag_04169, RS_AP_00134

[

Kind:	function	
Header file:	<code>#include "<data-identifier-directory-path>/{<data-identifier-shortname-lower>}.h"</code>	
Scope:	<code>class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}</code>	
Syntax:	<code>virtual ~{<data-identifier-interface-name-6>} ();</code>	
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Destruction of a <code>DataIdentifier</code>	
Descriptors:	<code>{<data-identifier-interface-name-6>}</code>	As per <code>{<data-identifier-interface-name-x>}</code> in [SWS_DM_00601]

]

8.29.3 Struct: {<data-identifier-name>}ReadOutput

[SWS_DM_00579] Definition of API class `{<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-name>}ReadOutput`

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

[

Kind:	struct	
Header file:	<code>#include "<data-identifier-directory-path>/{<data-identifier-shortname-lower>}.h"</code>	





Forwarding header file:	#include "<data-identifier-directory-path>/{<data-identifier-shortname-lower>}/{<data-identifier-shortname-lower>}_fwd.h"	
Scope:	class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}	
Symbol:	{<data-identifier-name>}ReadOutput	
Syntax:	struct {<data-identifier-name>}ReadOutput {...};	
Description:	Structure wrapping the <i>out</i> arguments for a ReadOutput	
Descriptors:	{<data-identifier-name>}	As per {<data-identifier-name>} in [SWS_DM_00640]
	{<data-identifier-out-args-members-ordered>}	Shown as "..." in Syntax. Each element is an <i>argument</i> in the ordered list of <i>arguments</i> , with <i>direction</i> == <i>out</i> , where [SWS_DM_80013] applies for each.

]

8.30 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 <i>DiagnosticRoutineInterface</i> 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 <i>DiagnosticRoutineInterface</i> . <i>shortName</i> converted to lower-case.
Example:	<pre>// 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)</pre>	

]

8.30.1 Namespaces

8.30.1.1 {<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.

]

8.30.2 Class: {<routine-interface-name-x>}

[SWS_DM_00604] Definition of API class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}

Status: DRAFT

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	class	
Port Interfaces:	DiagnosticRoutineInterface	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Forwarding header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	namespace {<routine-interface-hierarchical-namespace-list>}	
Symbol:	{<routine-interface-name-x>}	
Syntax:	class {<routine-interface-name-x>} { ... };	
Description:	DiagnosticRoutineInterface class	
Descriptors:	{<routine-interface-name-x>}	The DiagnosticRoutineInterface.shortName

]

8.30.2.1 Public Member Variables

8.30.2.1.1 {<routine-interface-request-result-out-arg-symbol>}

[SWS_DM_80005] Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::{<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-x>}	
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 <i>out</i> argument in an <i>RequestResultOutput</i> .	
Descriptors:	{<routine-interface-request-result-out-arg-type>}	The <i>ClientServerOperation.argument.type</i> , mapped to a C++ data type according to [27].
	{<routine-interface-request-result-out-arg-symbol>}	Symbol name of the <i>struct</i> element as given by <i>ClientServerOperation.ArgumentDataPrototype.shortName</i>

]

8.30.2.1.2 {<routine-interface-start-out-arg-symbol>}

[SWS_DM_80003] Definition of API variable {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::{<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-x>}	
Symbol:	{<routine-interface-start-out-arg-symbol>}	
Type:	{<routine-interface-start-out-arg-type>}	





Syntax:	<code>{<routine-interface-start-out-arg-type> <routine-interface-start-out-arg-symbol>;</code>	
Description:	Member declaration representing an <i>out</i> argument in an <i>StartOutput</i> .	
Descriptors:	<code>{<routine-interface-start-out-arg-type></code>	The <i>ClientServerOperation.argument.type</i> , mapped to a C++ data type according to [27].
	<code>{<routine-interface-start-out-arg-symbol></code>	Symbol name of the <i>struct</i> element as given by <i>ClientServerOperation.ArgumentDataPrototype.shortName</i>

]

8.30.2.1.3 {<routine-interface-stop-out-arg-symbol>}

[SWS_DM_80004] Definition of API variable `{<routine-interface-hierarchical-namespace-list>::{<routine-interface-name-x>>::
{<routine-interface-stop-out-arg-symbol>}`

Status: DRAFT

Upstream requirements: RS_Diag_04224, RS_Diag_04169

[

Kind:	variable	
Header file:	<code>#include "{<routine-interface-directory-path>}/{ <routine-interface-shortname-lower>}.h"</code>	
Scope:	<code>class {<routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>}</code>	
Symbol:	<code>{<routine-interface-stop-out-arg-symbol></code>	
Type:	<code>{<routine-interface-stop-out-arg-type></code>	
Syntax:	<code>{<routine-interface-stop-out-arg-type> <routine-interface-stop-out-arg-symbol>;</code>	
Description:	Member declaration representing an <i>out</i> argument in an <i>StopOutput</i> .	
Descriptors:	<code>{<routine-interface-stop-out-arg-type></code>	The <i>ClientServerOperation.argument.type</i> , mapped to a C++ data type according to [27].
	<code>{<routine-interface-stop-out-arg-symbol></code>	Symbol name of the <i>struct</i> element as given by <i>ClientServerOperation.ArgumentDataPrototype.shortName</i>

]

8.30.2.2 Public Member Functions

8.30.2.2.1 Member Functions

8.30.2.2.1.1 Offer

[SWS_DM_00594] Definition of API function {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::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-x>}	
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
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

]

8.30.2.2.1.2 RequestResult

[SWS_DM_00593] Definition of API function {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::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-x>}	
Syntax:	virtual ara::core::Future< {<routine-interface-name-l2>}RequestResult Output > 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) noexcept=0;	





Parameters (in):	{<routine-interface-request-result-in-arg-symbol-0toN>}	The symbol name of method <code>argument[0..N]</code> in the ordered list of method arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.ArgumentDataPrototype[0..N].shortName</code>
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< {<routine-interface-name-12>} RequestResultOutput ></code>	- If successful: an <code>ara::core::Future</code> containing an <code>RequestResult</code> object as per [SWS_DM_00583] - If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>ara::diag::DiagUdsNrcErrc</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Provision of a RequestResult RoutineControl (<code>DiagnosticRoutineInterface</code> in the role <code>requestResult</code>).	
Descriptors:	{<routine-interface-name-12>}	Name of a DiagnosticRoutine created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticRoutineInterface</code> in the role <code>requestResult</code> .
	{<derived-type>}	The C++ data type shall be derived according to the following rules: - If the (<code>CppImplementationDataType.category == VALUE</code>) or a (<code>CppImplementationDataType.category == TYPE_REFERENCE</code> which transitively type-resolves to a <code>CppImplementationDataType.category == VALUE</code>) and the <code>CppImplementationDataType.shortName</code> is either: <code>int8_t</code>: as per [SWS_APT_00001] <code>int16_t</code>: as per [SWS_APT_00004] <code>int32_t</code>: as per [SWS_APT_00007] <code>int64_t</code>: as per [SWS_APT_00010] <code>uint8_t</code>: as per [SWS_APT_00022] <code>uint16_t</code>: as per [SWS_APT_00025] <code>uint32_t</code>: as per [SWS_APT_00028] <code>uint64_t</code>: as per [SWS_APT_00031] <code>bool</code>: as per [SWS_APT_00049] <code>float</code>: as per [SWS_APT_00043] <code>double</code>: as per [SWS_APT_00046] in [28], then {<derived-type>} shall be unchanged, i.e. shall be the mapped C++ data type according to [27]. - Otherwise {<derived-type>} shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [29]), to the mapped C++ data type according to [27].





	{<routine-interface-request-result-in-arg-derived-type-0toN>}	The data type of method <code>argument[0..N]</code> in the ordered list of method arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.argument[0..N].type</code> , derived as per {<derived-type>}
Example:	<pre>// Example: virtual ara::core::Future<SomeRoutineInterfaceRequestResultOutput> RequestResult(// {<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::furtherthertype_t& inArgN, const ara::diag::MetaInfo& metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept = 0;</pre>	

]

8.30.2.2.1.3 Start

[SWS_DM_00591] Definition of API function {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::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-x>}	
Syntax:	<pre>virtual ara::core::Future< {<routine-interface-name-8>}StartOutput > Start ({<routine-interface-start-in-arg-derived-type-0toN>} {<routine-interface-start-in-arg-symbol-0toN>}, const ara::diag::Meta Info &metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	{<routine-interface-start-in-arg-symbol-0toN>}	The symbol name of method <code>argument[0..N]</code> in the ordered list of method arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.ArgumentDataPrototype[0..N].shortName</code>
	metaInfo	MetaInfo of the request.





	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< { <routine-interface-name-8> }StartOutput ></code>	- If successful: an <code>ara::core::Future</code> containing an <code>StartOutput</code> object as per [SWS_DM_00581] - If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>ara::diag::DiagUdsNrcErrc</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Provision of a Start RoutineControl (<code>DiagnosticRoutineInterface</code> in the role <code>start</code>).	
Descriptors:	<code>{<routine-interface-name-8>}</code>	Name of a <code>DiagnosticRoutine</code> created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticRoutineInterface</code> in the role <code>start</code> .
	<code>{<derived-type>}</code>	The C++ data type shall be derived according to the following rules: - If the <code>(CppImplementationDataType.category == VALUE)</code> or a <code>(CppImplementationDataType.category == TYPE_REFERENCE)</code> which transitively type-resolves to a <code>CppImplementationDataType.category == VALUE</code> and the <code>CppImplementationDataType.shortName</code> is either: <code>int8_t</code>: as per [SWS_APT_00001] <code>int16_t</code>: as per [SWS_APT_00004] <code>int32_t</code>: as per [SWS_APT_00007] <code>int64_t</code>: as per [SWS_APT_00010] <code>uint8_t</code>: as per [SWS_APT_00022] <code>uint16_t</code>: as per [SWS_APT_00025] <code>uint32_t</code>: as per [SWS_APT_00028] <code>uint64_t</code>: as per [SWS_APT_00031] <code>bool</code>: as per [SWS_APT_00049] <code>float</code>: as per [SWS_APT_00043] <code>double</code>: as per [SWS_APT_00046] in [28], then <code>{<derived-type>}</code> shall be unchanged, i.e. shall be the mapped C++ data type according to [27]. - Otherwise <code>{<derived-type>}</code> shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [29]), to the mapped C++ data type according to [27].
	<code>{<routine-interface-start-in-arg-derived-type-0toN>}</code>	The data type of method <code>argument[0..N]</code> in the ordered list of <code>method</code> arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.argument[0..N].type</code> , derived as per <code>{<derived-type>}</code>



**Example:**

```
// Example:
virtual ara::core::Future<SomeRoutineInterfaceStartOutput>
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
) noexcept = 0;
```

]

8.30.2.2.1.4 Stop

[SWS_DM_00592] Definition of API function {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::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-x>}	
Syntax:	virtual ara::core::Future< {<routine-interface-name-l0>}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) noexcept=0;	
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 given by ClientServerOperation.ArgumentDataPrototype[0..N].shortName
	metaInfo	MetaInfo of the request.





	cancellationHandler	This parameter is used to • Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. • Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	<code>ara::core::Future< { <routine-interface-name-10> >StopOutput ></code>	• If successful: an <code>ara::core::Future</code> containing an <code>StopOutput</code> object as per [SWS_DM_00582] • If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>ara::diag::DiagUdsNrcErrc</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	Please use an applicable NRC, as defined in ISO 14229-1, using <code>DiagUdsNrcErrc</code> .	
Description:	Provision of a Stop RoutineControl (<code>DiagnosticRoutineInterface</code> in the role <code>stop</code>).	
Descriptors:	<code>{<routine-interface-name-10> >}</code>	Name of a <code>DiagnosticRoutine</code> created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticRoutineInterface</code> in the role <code>stop</code> .
	<code>{<derived-type>}</code>	The C++ data type shall be derived according to the following rules: 1. If the <code>(CppImplementationDataType.category == VALUE)</code> or a <code>(CppImplementationDataType.category == TYPE_REFERENCE)</code> which transitively type-resolves to a <code>CppImplementationDataType.category == VALUE)</code> and the <code>CppImplementationDataType.shortName</code> is either: • <code>int8_t</code>: as per [SWS_APT_00001] • <code>int16_t</code>: as per [SWS_APT_00004] • <code>int32_t</code>: as per [SWS_APT_00007] • <code>int64_t</code>: as per [SWS_APT_00010] • <code>uint8_t</code>: as per [SWS_APT_00022] • <code>uint16_t</code>: as per [SWS_APT_00025] • <code>uint32_t</code>: as per [SWS_APT_00028] • <code>uint64_t</code>: as per [SWS_APT_00031] • <code>bool</code>: as per [SWS_APT_00049] • <code>float</code>: as per [SWS_APT_00043] • <code>double</code>: as per [SWS_APT_00046] in [28], then <code>{<derived-type>}</code> shall be unchanged, i.e. shall be the mapped C++ data type according to [27]. 2. Otherwise <code>{<derived-type>}</code> shall be a const-qualified lvalue reference (see [basic.type.qualifier], [basic.lval] in [29]), to the mapped C++ data type according to [27].
	<code>{<routine-interface-stop-in-arg-derived-type-0toN>}</code>	The data type of method <code>argument[0..N]</code> in the ordered list of <code>method</code> arguments, with <code>ClientServerOperation.direction == in</code> given by <code>ClientServerOperation.argument[0..N].type</code> , derived as per <code>{<derived-type>}</code>



**Example:**

```
// Example:
virtual ara::core::Future<SomeRoutineInterfaceStopOutput>
Stop(
    // {<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
) noexcept = 0;
```

8.30.2.2.1.5 StopOffer

[SWS_DM_00595] Definition of API function {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::StopOffer

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	function
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"
Scope:	class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}
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

]

8.30.2.2.1.6 operator=

[SWS_DM_01668] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::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-x>}	
Syntax:	{<routine-interface-name-5>} & operator= ({<routine-interface-name-5>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<routine-interface-name-5>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

]

8.30.2.2.1.7 operator=

[SWS_DM_01667] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::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-x>}	
Syntax:	{<routine-interface-name-3>} & operator= (const {<routine-interface-name-3>} &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	{<routine-interface-name-3>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

]

8.30.2.2.1.8 {<routine-interface-name-1>}

[SWS_DM_00589] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::{<routine-interface-name-1>}

Upstream requirements: RS_AP_00137, RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173

Kind:	function	
Header file:	#include "{<routine-interface-directory-path>}/{<routine-interface-shortname-lower>}.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}	
Syntax:	{<routine-interface-name-1>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Construct a DiagnosticRoutine from an ara::core::InstanceSpecifier	
Descriptors:	{<routine-interface-name-1>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

8.30.2.2.1.9 {<routine-interface-name-2>}

[SWS_DM_01669] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::{<routine-interface-name-2>}

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-x>}	
Syntax:	{<routine-interface-name-2>} (const {<routine-interface-name-2> &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	{<routine- interface-name-2>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

]

8.30.2.2.1.10 {<routine-interface-name-4>}

[SWS_DM_01670] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::
{<routine-interface-name-4>}

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-x>}	
Syntax:	{<routine-interface-name-4>} ({<routine-interface-name-4> &&other)=delete;	
Description:	Move constructor	
Descriptors:	{<routine- interface-name-4>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

]

8.30.2.2.1.11 ~{<routine-interface-name-6>}

[SWS_DM_00590] Definition of API function {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}::~~
{<routine-interface-name-6>}

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-x>}	





Syntax:	virtual ~{<routine-interface-name-6>} () noexcept;	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Destruction of a DiagnosticRoutine	
Descriptors:	{<routine-interface-name-6>}	As per {<routine-interface-name-x>} in [SWS_DM_00604]

]

8.30.3 Struct: {<routine-interface-name-11>}RequestResultOutput

[SWS_DM_00583] Definition of API class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::
{<routine-interface-name-11>}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>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}	
Symbol:	{<routine-interface-name-11>}RequestResultOutput	
Syntax:	struct {<routine-interface-name-11>}RequestResultOutput {...};	
Description:	Structure wrapping the <i>out</i> arguments for a RequestResult	
Descriptors:	{<routine-interface-name-11>}	As per {<routine-interface-name>} in [SWS_DM_00593]
	{<routine-interface-request-result-out-args-members-ordered>}	Shown as "..." in Syntax. Each element is an argument in the ordered list of arguments , with <i>direction</i> == <i>out</i> , where [SWS_DM_80005] applies for each.

]

8.30.4 Struct: {<routine-interface-name-6>}StartOutput

[SWS_DM_00581] Definition of API class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::
{<routine-interface-name-6>}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>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}	
Symbol:	{<routine-interface-name-6>}StartOutput	
Syntax:	struct {<routine-interface-name-6>}StartOutput {...};	
Description:	Structure wrapping the <i>out</i> arguments for a <i>Start</i>	
Descriptors:	{<routine-interface-name-6>}	As per {<routine-interface-name-x>} in [SWS_DM_00591]
	{<routine-interface-start-out-args-members-ordered>}	Shown as "..." in Syntax. Each element is an <i>argument</i> in the ordered list of <i>arguments</i> , with <i>direction</i> == <i>out</i> , where [SWS_DM_80003] applies for each.

]

8.30.5 Struct: {<routine-interface-name-9>}StopOutput

[SWS_DM_00582] Definition of API class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}::
{<routine-interface-name-9>}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>}/{<routine-interface-shortname-lower>}_fwd.h"	
Scope:	class {<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name-x>}	
Symbol:	{<routine-interface-name-9>}StopOutput	
Syntax:	struct {<routine-interface-name-9>}StopOutput {...};	





Description:	Structure wrapping the <i>out</i> arguments for a Stop	
Descriptors:	{<routine-interface-name-9>}	As per {<routine-interface-name-9>} in [SWS_DM_00592]
	{<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 , where [SWS_DM_80004] applies for each.

]

8.31 Diagnostic DoIP-related APIs

8.32 Header: ara/diag/doip_activationline.h

8.32.1 Class: DoIPActivationLine

[SWS_DM_00830] Definition of API class ara::diag::DoIPActivationLine

Upstream requirements: [RS_Diag_04242](#)

[

Kind:	class
Port Interfaces:	DiagnosticDoIPActivationLineInterface
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.

]

8.32.1.1 Public Member Functions

8.32.1.1.1 Special Member Functions

8.32.1.1.1.1 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:	<code>class ara::diag::DoIPActivationLine</code>
Syntax:	<code>DoIPActivationLine (DoIPActivationLine &&) noexcept=delete;</code>
Description:	Move constructor of DoIPActivationLine.

]

8.32.1.1.1.2 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:	<code>class ara::diag::DoIPActivationLine</code>
Syntax:	<code>DoIPActivationLine & operator= (DoIPActivationLine &&)=delete;</code>
Description:	Move assignment operator of DoIPActivationLine.

]

8.32.1.1.1.3 Destructor

[SWS_DM_00832] Definition of API function ara::diag::DoIPActivationLine::~DoIPActivationLineUpstream requirements: [RS_Diag_04242](#), [RS_AP_00134](#)

[

Kind:	function
Header file:	#include "ara/diag/doip_activationline.h"
Scope:	class ara::diag::DoIPActivationLine
Syntax:	<code>virtual ~DoIPActivationLine () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructor of DoIPActivationLine.

]

8.32.1.1.2 Constructors

8.32.1.1.2.1 DoIPActivationLine(const ara::core::InstanceSpecifier&)

[SWS_DM_00831] Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLineUpstream requirements: [RS_Diag_04242](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/doip_activationline.h"	
Scope:	class ara::diag::DoIPActivationLine	
Syntax:	<code>explicit DoIPActivationLine (const ara::core::InstanceSpecifier &specifier) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a 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.
	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.
	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.
Description:	Constructor of DoIPActivationLine.	

]

8.32.1.1.2.2 DoIPActivationLine(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:	<code>class ara::diag::DoIPActivationLine</code>
Syntax:	<code>DoIPActivationLine (DoIPActivationLine &)=delete;</code>
Description:	DoIPActivationLine shall be a single not copy-able instance.

]

8.32.1.1.3 Member Functions

8.32.1.1.3.1 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:	<code>class ara::diag::DoIPActivationLine</code>
Syntax:	<code>virtual ara::core::Future< bool > GetActivationLineState () noexcept=0;</code>
Return value:	<code>ara::core::Future< bool ></code> TRUE in case the activation line is active, else FALSE.
Exception Safety:	exception safe
Thread Safety:	thread-safe
Errors:	This function does not specify any standardized errors.
Description:	Called to get the current activation line state.

]

8.32.1.1.3.2 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	
Errors:	This function does not specify any standardized errors.	
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 DoIPNetworkConfiguration with property isActivationLineDependent = 'FALSE', this is an error!	

8.32.1.1.3.3 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics The service is already offered.
Description:	This Offer will enable the DM to listen to activation line state changes.	

8.32.1.1.3.4 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.

]

8.32.1.1.3.5 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:	void UpdateActivationLineState (bool activation_line_state);
Parameters (in):	activation_line_state TRUE in case the activation line is active, else FALSE
Return value:	None
Exception Safety:	not exception safe
Thread Safety:	not thread-safe
Description:	Called to update current activation line state.

]

8.32.1.1.3.6 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.

]

8.33 Header: ara/diag/doip_entity_identification.h

8.33.1 Class: DoIPEntityIdentification

[SWS_DM_01362] Definition of API class ara::diag::DoIPEntityIdentification

Upstream requirements: [RS_Diag_00026](#)

[

Kind:	class
Port Interfaces:	DiagnosticDoIPEntityIdentificationInterface
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:	<code>class DoIPEntityIdentification {...};</code>
Description:	DoIPEntityIdentification class to obtain the EID if required.

]

8.33.1.1 Public Member Functions

8.33.1.1.1 Special Member Functions

8.33.1.1.1.1 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:	<code>class ara::diag::DoIPEntityIdentification</code>
Syntax:	<code>virtual ~DoIPEntityIdentification () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPEntityIdentification.

]

8.33.1.1.2 Constructors

8.33.1.1.2.1 DoIPEntityIdentification

[SWS_DM_01364] Definition of API function `ara::diag::DoIPEntityIdentification::DoIPEntityIdentification`

Upstream requirements: [RS_AP_00137](#), [RS_Diag_00026](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/doip_entity_identification.h"	
Scope:	<code>class ara::diag::DoIPEntityIdentification</code>	
Syntax:	<code>explicit DoIPEntityIdentification (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticDoIPEntityIdentificationInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent)..
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.





	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.
	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.
Description:		Constructor of DoIPEntityIdentification.

8.33.1.1.3 Member Functions

8.33.1.1.3.1 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 <code>ara::diag::DoIPEntityIdentification</code>	
Syntax:	<code>virtual ara::core::Future< EntityId > GetEntityId () noexcept=0;</code>	
Return value:	<code>ara::core::Future< EntityId ></code>	Entity identification
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	Called to get the EID for the DoIP protocol.	

8.33.1.1.3.2 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:	<code>class ara::diag::DoIPEntityIdentification</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is 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_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:	<code>class ara::diag::DoIPEntityIdentification</code>	
Syntax:	<code>void StopOffer () noexcept;</code>	
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

8.33.2 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:	<code>class ara::diag::DoIPEntityIdentification</code>
Symbol:	EntityId
Syntax:	<code>struct EntityId {...};</code>
Description:	Entity Identification value (EID) as defined in ISO 13400-2 [2] .

]

8.33.2.1 Public Member Variables

8.33.2.1.1 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:	<code>struct ara::diag::DoIPEntityIdentification::EntityId</code>
Symbol:	entityIdentification
Type:	<code>ara::core::Array< std::uint8_t, 6 ></code>
Syntax:	<code>ara::core::Array<std::uint8_t, 6> entityIdentification;</code>
Description:	Entity Identification value.

]

8.34 Header: ara/diag/doip_group_identification.h

8.34.1 Class: DoIPGroupIdentification

[SWS_DM_00720] Definition of API class ara::diag::DoIPGroupIdentification

Upstream requirements: [RS_Diag_00026](#)

[

Kind:	class
Port Interfaces:	DiagnosticDoIPGroupIdentificationInterface
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:	<code>class DoIPGroupIdentification {...};</code>
Description:	DoIPGroupIdentificationInterface.

]

8.34.1.1 Public Member Functions

8.34.1.1.1 Special Member Functions

8.34.1.1.1.1 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:	<code>DoIPGroupIdentification (DoIPGroupIdentification &&) noexcept=delete;</code>
Description:	Move constructor of DoIPGroupIdentification.

]

8.34.1.1.1.2 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:	<code>class ara::diag::DoIPGroupIdentification</code>
Syntax:	<code>DoIPGroupIdentification & operator= (DoIPGroupIdentification &&) = delete;</code>
Description:	Move assignment operator of DoIPGroupIdentification.

]

8.34.1.1.1.3 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:	<code>class ara::diag::DoIPGroupIdentification</code>
Syntax:	<code>virtual ~DoIPGroupIdentification () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPGroupIdentification.

]

8.34.1.1.2 Constructors

8.34.1.1.2.1 DoIPGroupIdentification(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_00722] Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification

Upstream requirements: [RS_AP_00137](#), [RS_Diag_00026](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

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 a PortPrototype of a DiagnosticDoIPGroupIdentificationInterface .
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	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.
	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.
Description:	Constructor of DoIPGroupIdentification.	

8.34.1.1.2.2 DoIPGroupIdentification(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:	<code>DoIPGroupIdentification (DoIPGroupIdentification &)=delete;</code>
Description:	DoIPGroupIdentification shall be a single not copy-able instance.

]

8.34.1.1.3 Member Functions

8.34.1.1.3.1 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:	<code>class ara::diag::DoIPGroupIdentification</code>	
Syntax:	<code>virtual ara::core::Future< GidStatus > GetGidStatus () noexcept=0;</code>	
Return value:	<code>ara::core::Future< GidStatus ></code>	group identification and state
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	Called to get the current GID state for the DoIP protocol.	

]

8.34.1.1.3.2 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:	<code>class ara::diag::DoIPGroupIdentification</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	





Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.34.1.1.3.3 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:	<code>class ara::diag::DoIPGroupIdentification</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM to this handler.

]

8.34.1.1.3.4 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:	<code>class ara::diag::DoIPGroupIdentification</code>
Syntax:	<code>DoIPGroupIdentification & operator= (DoIPGroupIdentification &)=delete;</code>
Description:	DoIPGroupIdentification shall be a single not assignable instance.

]

8.34.2 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.

]

8.35 Header: ara/diag/doip_power_mode.h

8.35.1 Non-Member Types

8.35.1.1 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:	std::uint8_t		
Syntax:	enum class PowerModeType : std::uint8_t {...};		
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 ISO 13400-2 [2].		

]

8.35.2 Class: DoIPPowerMode

[SWS_DM_00731] Definition of API class ara::diag::DoIPPowerMode

Upstream requirements: [RS_Diag_00080](#)

[

Kind:	class
Port Interfaces:	DiagnosticDoIPPowerModeInterface
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.

]

8.35.2.1 Public Member Functions

8.35.2.1.1 Special Member Functions

8.35.2.1.1.1 Move Constructor

[SWS_DM_01646] 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 &&) noexcept=delete;
Description:	Move constructor of DoIPPowerMode.

]

8.35.2.1.1.2 Move Assignment Operator

[SWS_DM_01644] 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:	<code>class ara::diag::DoIPPowerMode</code>
Syntax:	<code>DoIPPowerMode & operator= (DoIPPowerMode &&)=delete;</code>
Description:	Move assignment operator of DoIPPowerMode.

]

8.35.2.1.1.3 Destructor

[SWS_DM_00733] Definition of API function `ara::diag::DoIPPowerMode::~~DoIPPowerMode`

Upstream requirements: [RS_AP_00134](#), [RS_Diag_00080](#)

[

Kind:	function
Header file:	#include "ara/diag/doip_power_mode.h"
Scope:	<code>class ara::diag::DoIPPowerMode</code>
Syntax:	<code>virtual ~DoIPPowerMode () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DoIPPowerMode .

]

8.35.2.1.2 Constructors

8.35.2.1.2.1 DoIPPowerMode(const ara::core::InstanceSpecifier&, ConcurrencyType)

[SWS_DM_00732] Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode

Upstream requirements: [RS_AP_00137](#), [RS_Diag_00080](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/doip_power_mode.h"	
Scope:	<code>class ara::diag::DoIPPowerMode</code>	
Syntax:	<code>explicit DoIPPowerMode (const ara::core::InstanceSpecifier &specifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticDoIPPowerModeInterface
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent)..
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.
	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.
	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.
Description:	Constructor of DoIPPowerMode.	

8.35.2.1.2.2 DoIPPowerMode(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:	<code>class ara::diag::DoIPPowerMode</code>	
Syntax:	<code>DoIPPowerMode (DoIPPowerMode &)=delete;</code>	





Description:	DoIPPowerMode shall be a single not copiable instance.

]

8.35.2.1.3 Member Functions

8.35.2.1.3.1 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:	<code>class ara::diag::DoIPPowerMode</code>	
Syntax:	<code>virtual ara::core::Future< PowerModeType > GetDoIPPowerMode () noexcept=0;</code>	
Return value:	<code>ara::core::Future< PowerModeType ></code>	current diagnostic power mode
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	Called to get the current Power Mode for the DoIP protocol.	

]

8.35.2.1.3.2 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:	<code>class ara::diag::DoIPPowerMode</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	





Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.35.2.1.3.3 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.

]

8.35.2.1.3.4 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.

]

8.36 Header: ara/diag/doip_trigger_vehicle_announcement.h

8.36.1 Class: DoIPTriggerVehicleAnnouncement

[SWS_DM_00820] Definition of API class ara::diag::DoIPTriggerVehicleAnnouncement

Upstream requirements: [RS_Diag_04242](#)

[

Kind:	class
Port Interfaces:	DiagnosticDoIPTriggerVehicleAnnouncementInterface
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.

]

8.36.1.1 Public Member Functions

8.36.1.1.1 Member Functions

8.36.1.1.1.1 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< DoIPTriggerVehicleAnnouncement > > GetDoIPTriggerVehicleAnnouncement () noexcept;	
Return value:	ara::core::Result< std::reference_wrapper< DoIPTriggerVehicleAnnouncement > >	DoIPTriggerVehicleAnnouncement object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Get DoIPTriggerVehicleAnnouncement interface from DM.	

]

8.36.1.1.1.2 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;	
Parameters (in):	networkInterfaceId	network interface id on which a vehicle announcement shall be sent.
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

8.36.1.2 Private Member Functions

8.36.1.2.1 Special Member Functions

8.36.1.2.1.1 Default Constructor

[SWS_DM_02100] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::DoIPTriggerVehicleAnnouncement

Upstream requirements: [RS_Diag_04242](#)

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	

8.36.1.2.1.2 Destructor

[SWS_DM_02101] Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::~~DoIPTriggerVehicleAnnouncement

Upstream requirements: [RS_Diag_04242](#)

[

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

]

8.37 Diagnostic Event-related APIs

8.38 Header: ara/diag/condition.h

8.38.1 Non-Member Types

8.38.1.1 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:	std::uint8_t	
Syntax:	enum class ConditionType : std::uint8_t {...};	
Values:	kConditionFalse	= 0x00
		condition is set to false
	kConditionTrue	= 0x01
		condition is set to true
Description:	Type for Condition status.	

]

8.38.2 Class: Condition

[SWS_DM_00711] Definition of API class ara::diag::Condition

Upstream requirements: [RS_Diag_04192](#)

[

Kind:	class
Port Interfaces:	DiagnosticConditionInterface
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.

]

8.38.2.1 Public Member Functions

8.38.2.1.1 Special Member Functions

8.38.2.1.1.1 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.

]

8.38.2.1.1.2 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:	<code>class ara::diag::Condition</code>
Syntax:	<code>Condition & operator= (Condition &&) = delete;</code>
Description:	Move assignment operator of Condition.

]

8.38.2.1.1.3 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:	<code>class ara::diag::Condition</code>
Syntax:	<code>~Condition () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class Condition .

]

8.38.2.1.2 Constructors

8.38.2.1.2.1 Condition(const ara::core::InstanceSpecifier&)

[SWS_DM_00712] Definition of API function ara::diag::Condition::Condition

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04192](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_ - AP_00172](#), [RS_AP_00173](#)

[

Kind:	function
Header file:	#include "ara/diag/condition.h"
Scope:	<code>class ara::diag::Condition</code>





Syntax:	explicit Condition (const ara::core::InstanceSpecifier &specifier) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticConditionInterface .
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.
	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.
	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.
Description:	Constructor of Condition Class.	

]

8.38.2.1.2.2 Condition(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.

]

8.38.2.1.3 Member Functions

8.38.2.1.3.1 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:	<code>class ara::diag::Condition</code>	
Syntax:	<code>ara::core::Future< ConditionType > GetCondition () noexcept;</code>	
Return value:	<code>ara::core::Future< ConditionType ></code>	the current condition
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get current condition.	

]

8.38.2.1.3.2 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:	<code>class ara::diag::Condition</code>	
Syntax:	<code>ara::core::Result< void > SetCondition (ConditionType condition) noexcept;</code>	
Parameters (in):	condition	current condition
Return value:	<code>ara::core::Result< void ></code>	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Set condition.	

]

8.38.2.1.3.3 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:	<code>class ara::diag::Condition</code>
Syntax:	<code>Condition & operator= (Condition &)=delete;</code>
Description:	Condition shall be a single not assignable instance.

]

8.39 Header: ara/diag/operation_cycle.h

8.39.1 Class: OperationCycle

[SWS_DM_00751] Definition of API class ara::diag::OperationCycle

Upstream requirements: [RS_Diag_04178](#)

[

Kind:	class
Port Interfaces:	DiagnosticOperationCycleInterface
Header file:	#include "ara/diag/operation_cycle.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	OperationCycle
Syntax:	<code>class OperationCycle final {...};</code>
Description:	DiagnosticOperationCycleInterface provides functionality for handling of operation cycles.

]

8.39.1.1 Public Member Types

8.39.1.1.1 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:	<code>#include "ara/diag/operation_cycle.h"</code>
Scope:	<code>class ara::diag::OperationCycle</code>
Symbol:	<code>OperationCycleSetNotifier</code>
Syntax:	<code>using OperationCycleSetNotifier = std::function<void(void)>;</code>
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the operation cycle is changed. .

]

8.39.1.2 Public Member Functions

8.39.1.2.1 Special Member Functions

8.39.1.2.1.1 Move Constructor

[SWS_DM_01599] Definition of API function `ara::diag::OperationCycle::OperationCycle`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/operation_cycle.h"</code>
Scope:	<code>class ara::diag::OperationCycle</code>
Syntax:	<code>OperationCycle (OperationCycle &&) noexcept=delete;</code>
Description:	Move constructor of OperationCycle.

]

8.39.1.2.1.2 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:	<code>class ara::diag::OperationCycle</code>
Syntax:	<code>OperationCycle & operator= (OperationCycle &&)=delete;</code>
Description:	Move assignment operator of OperationCycle.

]

8.39.1.2.1.3 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:	<code>class ara::diag::OperationCycle</code>
Syntax:	<code>~OperationCycle () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticOperationCycleInterface .

]

8.39.1.2.2 Constructors

8.39.1.2.2.1 OperationCycle(const ara::core::InstanceSpecifier&)

[SWS_DM_00752] Definition of API function ara::diag::OperationCycle::OperationCycle

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04178](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

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 a PortPrototype of a 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.
	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.
	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.
Description:	Constructor for DiagnosticOperationCycleInterface.	

8.39.1.2.2.2 OperationCycle(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.	

8.39.1.2.3 Member Functions

8.39.1.2.3.1 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:	<code>class ara::diag::OperationCycle</code>	
Syntax:	<code>ara::core::Result< void > RestartOperationCycle () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Trigger to restart the Operation Cycle .	

]

8.39.1.2.3.2 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:	<code>class ara::diag::OperationCycle</code>	
Syntax:	<code>ara::core::Result< void > SetNotifier (OperationCycleSetNotifier notifier) noexcept;</code>	
Parameters (in):	notifier	notifier function
Return value:	<code>ara::core::Result< void ></code>	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.39.1.2.3.3 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.

]

8.40 Header: ara/diag/monitor.h

8.40.1 Class: Monitor

[SWS_DM_00542] Definition of API class ara::diag::Monitor

Upstream requirements: [RS_Diag_04179](#)

[

Kind:	class
Port Interfaces:	DiagnosticMonitorInterface
Header file:	#include "ara/diag/monitor.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	Monitor
Syntax:	<code>class Monitor final {...};</code>
Description:	Class to implement operations on diagnostic Monitor interface.

]

8.40.1.1 Public Member Functions

8.40.1.1.1 Special Member Functions

8.40.1.1.1.1 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:	<code>class ara::diag::Monitor</code>
Syntax:	<code>Monitor (Monitor &&) noexcept=delete;</code>
Description:	Move constructor of Monitor.

]

8.40.1.1.1.2 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:	<code>class ara::diag::Monitor</code>
Syntax:	<code>Monitor & operator= (Monitor &&)=delete;</code>
Description:	Move assignment operator of Monitor.

]

8.40.1.1.2 Constructors

8.40.1.1.2.1 Monitor(const ara::core::InstanceSpecifier&, InitMonitorNotifier)

[SWS_DM_01972] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT

Upstream requirements: [RS_AP_00121](#), [RS_AP_00137](#), [RS_Diag_04179](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	<code>class ara::diag::Monitor</code>	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, <code>InitMonitorNotifier</code> initMonitor) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a <code>PortPrototype</code> of a <code>DiagnosticMonitorInterface</code> .
	initMonitor	Possibility to register an InitMonitor callback
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Violations:	<code>InstanceSpecifierMappingIntegrityViolation</code>	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.
	<code>PortInterfaceMappingViolation</code>	A <code>PortPrototype</code> that is referenced by a <code>DiagnosticMonitorPortMapping</code> needs to be typed by a <code>DiagnosticMonitorInterface</code> .
	<code>ProcessMappingViolation</code>	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.
	<code>InstanceSpecifierAlreadyInUseViolation</code>	Violation message that is sent in case a constructor in the ara framework was called with an InstanceSpecifier already in use in this process.
	<code>InvalidDebounceAlgorithmViolation</code>	Mismatching Debounce Algorithm in DEXT i.e <code>DiagnosticEventToDebounceAlgorithmMapping</code> shall be mapped to <code>DiagnosticEventToDebounceAlgorithmMapping</code> . <code>DiagnosticDebounceAlgorithmProps</code> which aggregates <code>DiagEventDebounceMonitorInternal</code> or <code>DiagnosticEventToDebounceAlgorithmMapping</code> shall not exist.
Description:	Monitor constructor for Monitors without debouncing.	

8.40.1.1.2.2 Monitor(const ara::core::InstanceSpecifier&, InitMonitorNotifier, CounterBased)

[SWS_DM_00549] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT

Upstream requirements: [RS_AP_00137](#), [RS_AP_00121](#), [RS_Diag_04179](#), [RS_Diag_04068](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	<code>class ara::diag::Monitor</code>	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, InitMonitorNotifier initMonitor, CounterBased defaultValues) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticMonitorInterface .
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
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.
	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.
	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.
	InvalidDebounceAlgorithmViolation	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.	

8.40.1.1.2.3 Monitor(const ara::core::InstanceSpecifier&, InitMonitorNotifier, TimeBased)

[SWS_DM_00550] Definition of API function ara::diag::Monitor::Monitor

Status: DRAFT

Upstream requirements: [RS_AP_00121](#), [RS_AP_00137](#), [RS_Diag_04179](#), [RS_Diag_04225](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	<code>class ara::diag::Monitor</code>	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, InitMonitorNotifier initMonitor, TimeBased defaultValues) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticMonitorInterface .
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
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.
	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.
	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.
	InvalidDebounceAlgorithmViolation	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.	

8.40.1.1.2.4 Monitor(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:	<code>class ara::diag::Monitor</code>
Syntax:	<code>Monitor (Monitor &)=delete;</code>
Description:	Monitor shall be a single not copy-able instance.

]

8.40.1.1.2.5 Monitor(const ara::core::InstanceSpecifier&, InitMonitorNotifier, std::function<std::int8_t(void)>)**[SWS_DM_00548] Definition of API function ara::diag::Monitor::Monitor***Status:* DRAFT*Upstream requirements:* [RS_AP_00121](#), [RS_AP_00137](#), [RS_Diag_04179](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/monitor.h"	
Scope:	<code>class ara::diag::Monitor</code>	
Syntax:	<code>Monitor (const ara::core::InstanceSpecifier &specifier, InitMonitorNotifier initMonitor, std::function< std::int8_t()> getFaultDetectionCounter) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticMonitorInterface .
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetectionCounter	Possibility to register a function to get the current FDC for this event.
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.
	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.
	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.





	InvalidDebounceAlgorithmViolation	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 ;	

]

8.40.1.1.3 Member Functions

8.40.1.1.3.1 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	With the Offer the application states that it is ready to receive and process initMonitor callouts.	

]

8.40.1.1.3.2 ReportMonitorAction

[SWS_DM_00543] Definition of API function [ara::diag::Monitor::ReportMonitorAction](#)

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:	UnexpectedMonitorActionHandlingViolation	<code>ara::diag::Monitor::ReportMonitorAction</code> is called with a parameter <code>action</code> value that is not supported by the used debouncing method.
Description:	Function to report the status information being relevant for error monitoring paths.	

]

8.40.1.1.3.3 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:	<code>class ara::diag::Monitor</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.40.1.1.3.4 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:	<code>class ara::diag::Monitor</code>
Syntax:	<code>Monitor & operator= (Monitor &)=delete;</code>
Description:	Monitor shall be a single not assignable instance.

]

8.41 Header: ara/diag/monitor_types.h

8.41.1 Non-Member Types

8.41.1.1 Type Alias: InitMonitorNotifier

[SWS_DM_02121] Definition of API type ara::diag::InitMonitorNotifier

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04179](#), [RS_AP_00121](#)

[

Kind:	type alias
Header file:	#include "ara/diag/monitor_types.h"
Scope:	namespace ara::diag
Symbol:	InitMonitorNotifier
Syntax:	using InitMonitorNotifier = std::function<void(InitMonitorReason)>;
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the monitor needs to be re-initialized.

]

8.41.1.2 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-initialized.	

]

8.41.1.3 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.
	kFdcThresholdReached	= 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.	

]

8.41.2 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:	<code>struct CounterBased {...};</code>
Description:	Represents the parameters for counter-based debouncing.

]

8.41.2.1 Public Member Variables

8.41.2.1.1 failedJumpValue

[SWS_DM_00625] Definition of API variable `ara::diag::CounterBased::failedJumpValue`

Upstream requirements: [RS_Diag_04068](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/monitor_types.h"</code>
Scope:	<code>struct ara::diag::CounterBased</code>
Symbol:	<code>failedJumpValue</code>
Type:	<code>std::int16_t</code>
Syntax:	<code>std::int16_t failedJumpValue;</code>
Description:	failed to jump value

]

8.41.2.1.2 failedStepsize

[SWS_DM_00623] Definition of API variable `ara::diag::CounterBased::failedStepsize`

Upstream requirements: [RS_Diag_04068](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/monitor_types.h"</code>
Scope:	<code>struct ara::diag::CounterBased</code>
Symbol:	<code>failedStepsize</code>
Type:	<code>std::uint16_t</code>
Syntax:	<code>std::uint16_t failedStepsize;</code>
Description:	Stepsize per pre-failed report.

]

8.41.2.1.3 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.

]

8.41.2.1.4 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

]

8.41.2.1.5 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.

]

8.41.2.1.6 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.

]

8.41.2.1.7 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

]

8.41.2.1.8 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

]

8.41.3 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.

]

8.41.3.1 Public Member Variables

8.41.3.1.1 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)

]

8.41.3.1.2 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)

]

8.42 Header: ara/diag/multiple_condition.h

8.42.1 Non-Member Types

8.42.1.1 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 condition.

]

8.42.2 Class: MultipleCondition

[SWS_DM_01730] Definition of API class ara::diag::MultipleCondition

Upstream requirements: [RS_Diag_04192](#)

[

Kind:	class
Port Interfaces:	DiagnosticMultipleConditionInterface
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.

]

8.42.2.1 Public Member Functions

8.42.2.1.1 Special Member Functions

8.42.2.1.1.1 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:	<code>class ara::diag::MultipleCondition</code>
Syntax:	<code>MultipleCondition (MultipleCondition &&) noexcept=delete;</code>
Description:	Move constructor of MultipleCondition.

]

8.42.2.1.1.2 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:	<code>class ara::diag::MultipleCondition</code>
Syntax:	<code>MultipleCondition & operator= (MultipleCondition &&)=delete;</code>
Description:	Move assignment operator of MultipleCondition.

]

8.42.2.1.1.3 Destructor

[SWS_DM_01737] Definition of API function ara::diag::MultipleCondition::~~MultipleConditionUpstream 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 .

]

8.42.2.1.2 Constructors

8.42.2.1.2.1 MultipleCondition(const ara::core::InstanceSpecifier&)

[SWS_DM_01727] Definition of API function ara::diag::MultipleCondition::MultipleConditionUpstream requirements: [RS_AP_00137](#), [RS_Diag_04192](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

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 a PortPrototype of a 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.
	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.
	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.





Description:	Constructor of MultipleCondition Class.

]

8.42.2.1.2.2 MultipleCondition(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:	<code>class ara::diag::MultipleCondition</code>
Syntax:	<code>MultipleCondition (MultipleCondition &)=delete;</code>
Description:	Copy constructor of MultipleCondition.

]

8.42.2.1.3 Member Functions

8.42.2.1.3.1 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:	<code>class ara::diag::MultipleCondition</code>	
Syntax:	<code>ara::core::Future< ConditionType > GetCondition (ConditionHandleType conditionHandle) noexcept;</code>	
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing





	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get condition state of the requested condition.	

]

8.42.2.1.3.2 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:	<code>class ara::diag::MultipleCondition</code>	
Syntax:	<code>ara::core::Result< void > SetCondition (ConditionHandleType condition Handle, ConditionType condition) noexcept;</code>	
Parameters (in):	conditionHandle	Identifies the requested condition
	condition	current condition
Return value:	<code>ara::core::Result< void ></code> Returns nothing or an error.	
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Set condition state of the requested condition.	

]

8.42.2.1.3.3 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:	<code>class ara::diag::MultipleCondition</code>	
Syntax:	<code>MultipleCondition & operator= (MultipleCondition &)=delete;</code>	





Description:	Copy assignment operator of MultipleCondition.

]

8.43 Header: ara/diag/multiple_event.h

8.43.1 Non-Member Types

8.43.1.1 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.

]

8.43.2 Class: MultipleEvent

[SWS_DM_01704] Definition of API class ara::diag::MultipleEvent

Upstream requirements: [RS_Diag_04151](#)

[

Kind:	class
Port Interfaces:	DiagnosticMultipleEventInterface
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.

]

8.43.2.1 Public Member Functions

8.43.2.1.1 Special Member Functions

8.43.2.1.1.1 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:	<code>class ara::diag::MultipleEvent</code>
Syntax:	<code>MultipleEvent (MultipleEvent &&) noexcept=delete;</code>
Description:	Move constructor of MultipleEvent.

]

8.43.2.1.1.2 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:	<code>class ara::diag::MultipleEvent</code>
Syntax:	<code>MultipleEvent & operator= (MultipleEvent &&)=delete;</code>
Description:	Move assignment operator of MultipleEvent.

]

8.43.2.1.1.3 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:	<code>class ara::diag::MultipleEvent</code>





Syntax:	<code>~MultipleEvent () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class MultipleEvent.

]

8.43.2.1.2 Constructors

8.43.2.1.2.1 MultipleEvent(MultipleEvent&)

[SWS_DM_02070] Definition of API function `ara::diag::MultipleEvent::MultipleEvent`

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/multiple_event.h"</code>
Scope:	<code>class ara::diag::MultipleEvent</code>
Syntax:	<code>MultipleEvent (MultipleEvent &)=delete;</code>
Description:	Copy constructor of MultipleEvent.

]

8.43.2.1.2.2 MultipleEvent(const ara::core::InstanceSpecifier&)

[SWS_DM_01705] Definition of API function `ara::diag::MultipleEvent::MultipleEvent`

Upstream requirements: [RS_Diag_04151](#), [RS_AP_00137](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	<code>#include "ara/diag/multiple_event.h"</code>	
Scope:	<code>class ara::diag::MultipleEvent</code>	
Syntax:	<code>explicit MultipleEvent (const ara::core::InstanceSpecifier &specifier) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticMultipleEventInterface .
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.
	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.
	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.
Description:	Constructor MultipleEvent class.	

8.43.2.1.3 Member Functions

8.43.2.1.3.1 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:	<code>ara::core::Future< std::uint32_t > GetDTCNumber (EventHandleType eventHandle, DTCFormatType dtcFormat) noexcept;</code>	
Parameters (in):	eventHandle	Identifies the requested event
	dtcFormat	Define DTC format type for the return value.
Return value:	<code>ara::core::Future< std::uint32_t ></code>	DTC number in respective DTCFormatType .
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
	DiagErrc::kNoSuchDTC	rollback_semantics
		No mapped DTC exists for the requested event.
Errors:	DiagErrc::kServiceNotAvailable	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the DTC-ID related to requested event.	

8.43.2.1.3.2 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
	DiagErrc::kServiceNotAvailable	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.	

]

8.43.2.1.3.3 GetEventStatus**[SWS_DM_01707] Definition of API function ara::diag::MultipleEvent::GetEventStatus**

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;	
Parameters (in):	eventHandle	Handle of the Event
Return value:	ara::core::Future< EventStatusByte >	Returns the current diagnostic event status
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor





Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
	DiagErrc::kServiceNot Available	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current status of the requested event.	

]

8.43.2.1.3.4 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:	<code>class ara::diag::MultipleEvent</code>	
Syntax:	<code>ara::core::Future< std::int8_t > GetFaultDetectionCounter (EventHandleType eventHandle) noexcept;</code>	
Parameters (in):	eventHandle	Identifies the requested event
Return value:	ara::core::Future< std::int8_t >	Return the current FaultDetectionCounter value.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
	DiagErrc::kServiceNot Available	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.	

]

8.43.2.1.3.5 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, EventStatusByteNotifier 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

8.43.2.1.3.6 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.	

8.44 Header: ara/diag/multiple_monitor.h

8.44.1 Non-Member Types

8.44.1.1 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 .

]

8.44.2 Class: MultipleMonitor

[SWS_DM_01694] Definition of API class ara::diag::MultipleMonitor

Upstream requirements: [RS_Diag_04179](#)

[

Kind:	class
Port Interfaces:	DiagnosticMultipleMonitorInterface
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.

]

8.44.2.1 Public Member Functions

8.44.2.1.1 Special Member Functions

8.44.2.1.1.1 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:	<code>class ara::diag::MultipleMonitor</code>
Syntax:	<code>MultipleMonitor (MultipleMonitor &&) noexcept=delete;</code>
Description:	Move constructor of MultiMonitor.

]

8.44.2.1.1.2 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:	<code>class ara::diag::MultipleMonitor</code>
Syntax:	<code>MultipleMonitor & operator= (MultipleMonitor &&)=delete;</code>
Description:	Move assignment operator of MultiMonitor.

]

8.44.2.1.2 Constructors

8.44.2.1.2.1 MultipleMonitor

[SWS_DM_01695] Definition of API function `ara::diag::MultipleMonitor::MultipleMonitor`

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04179](#), [RS_AP_00121](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/multiple_monitor.h"	
Scope:	<code>class ara::diag::MultipleMonitor</code>	
Syntax:	<code>MultipleMonitor (const ara::core::InstanceSpecifier &specifier) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a 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.
	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.
	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.
Description:	MultipleMonitor constructor allows to report status of more than one monitor. using <code>ara::diag::CounterBased</code> ; using <code>ara::diag::TimeBased</code> ;	

8.44.2.1.2.2 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:	<code>class ara::diag::MultipleMonitor</code>	
Syntax:	<code>MultipleMonitor (MultipleMonitor &)=delete;</code>	
Description:	Copy constructor of MultiMonitor.	

8.44.2.1.3 Member Functions

8.44.2.1.3.1 ConfigureMonitor(MonitorHandleType, InitMonitorNotifier, Counter Based)

[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, InitMonitorNotifier 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

]

8.44.2.1.3.2 ConfigureMonitor(MonitorHandleType, InitMonitorNotifier, Time Based)

[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, InitMonitorNotifier initMonitor, TimeBased defaultValues) 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

┌

8.44.2.1.3.3 ConfigureMonitor(MonitorHandleType, InitMonitorNotifier)

[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, InitMonitorNotifier initMonitor) noexcept;	
Parameters (in):	monitorHandle	Handle of the monitor
	initMonitor	Possibility to register an InitMonitor callback.
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

┌

8.44.2.1.3.4 ConfigureMonitor(MonitorHandleType, InitMonitorNotifier, std::function<std::int8_t(void)>)

[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, InitMonitorNotifier initMonitor, std::function< std::int8_t()> getFaultDetectionCounter) noexcept;	
Parameters (in):	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetectionCounter	Possibility to register a function to get the current FDC for this event.
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

]

8.44.2.1.3.5 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.44.2.1.3.6 ReportMonitorAction**[SWS_DM_01698] Definition of API function ara::diag::MultipleMonitor::ReportMonitorAction***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 non-thread safe is determined by the ConcurrencyType given in the constructor
Description:	Function to report the status information being relevant for error monitoring paths. Calls with invalid monitorHandle are ignored.	

]

8.44.2.1.3.7 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.	

]

8.44.2.1.3.8 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:	<code>class ara::diag::MultipleMonitor</code>
Syntax:	<code>MultipleMonitor & operator= (MultipleMonitor &)=delete;</code>
Description:	Copy assignment operator of MultiMonitor.

]

8.45 Header: ara/diag/indicator.h

8.45.1 Non-Member Types

8.45.1.1 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:	std::uint8_t	
Syntax:	enum class IndicatorStatusType : std::uint8_t {...};	
Values:	kOff	= 0x00 Indicator off mode {default}.
	kOnDemand	= 0x06 Indicator on-demand mode.
Description:	Represents the state of an indicator.	

]

8.45.2 Class: Indicator

[SWS_DM_00741] Definition of API class ara::diag::Indicator

Upstream requirements: [RS_Diag_04204](#)

Kind:	class
Port Interfaces:	DiagnosticIndicatorInterface
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.

8.45.2.1 Public Member Types

8.45.2.1.1 Type Alias: IndicatorStatusTypeNotifier

[SWS_DM_02129] Definition of API type ara::diag::Indicator::IndicatorStatusTypeNotifier

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04204](#)

Kind:	type alias
Header file:	#include "ara/diag/indicator.h"
Scope:	class ara::diag::Indicator
Symbol:	IndicatorStatusTypeNotifier
Syntax:	using IndicatorStatusTypeNotifier = std::function<void(IndicatorStatusType)>;
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the indicator state is updated.

8.45.2.2 Public Member Functions

8.45.2.2.1 Special Member Functions

8.45.2.2.1.1 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:	<code>class ara::diag::Indicator</code>
Syntax:	<code>Indicator (Indicator &&) noexcept=delete;</code>
Description:	Move constructor of Indicator.

]

8.45.2.2.1.2 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:	<code>class ara::diag::Indicator</code>
Syntax:	<code>Indicator & operator= (Indicator &&)=delete;</code>
Description:	Move assignment operator of Indicator.

]

8.45.2.2.1.3 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:	<code>class ara::diag::Indicator</code>
Syntax:	<code>~Indicator () noexcept;</code>
Exception Safety:	exception safe

▽



Thread Safety:	not thread-safe
Description:	Destructor of DiagnosticIndicatorInterface.

]

8.45.2.2.2 Constructors

8.45.2.2.2.1 Indicator(const ara::core::InstanceSpecifier&)

[SWS_DM_00742] Definition of API function ara::diag::Indicator::Indicator

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04204](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

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 a PortPrototype of a DiagnosticIndicatorInterface .
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.
	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.
	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.
Description:	Constructor for DiagnosticIndicatorInterface.	

]

8.45.2.2.2 Indicator(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:	<code>class ara::diag::Indicator</code>
Syntax:	<code>Indicator (Indicator &)=delete;</code>
Description:	Indicator shall be a single not copy-able instance.

8.45.2.2.3 Member Functions

8.45.2.2.3.1 GetIndicatorState

[SWS_DM_00744] Definition of API function ara::diag::Indicator::GetIndicatorState

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04204](#)

Kind:	function	
Header file:	#include "ara/diag/indicator.h"	
Scope:	<code>class ara::diag::Indicator</code>	
Syntax:	<code>ara::core::Future< IndicatorStatusType > GetIndicatorState () noexcept;</code>	
Return value:	<code>ara::core::Future< IndicatorStatusType ></code>	Returns the current IndicatorState.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get current IndicatorState.	

8.45.2.2.3.2 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:	<code>class ara::diag::Indicator</code>	
Syntax:	<code>ara::core::Result< void > SetNotifier (IndicatorStatusTypeNotifier notifier) noexcept;</code>	
Parameters (in):	notifier	notifier function
Return value:	<code>ara::core::Result< void ></code>	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.45.2.2.3.3 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:	<code>class ara::diag::Indicator</code>	
Syntax:	<code>Indicator & operator= (Indicator &)=delete;</code>	
Description:	Indicator shall be a single not assign-able instance.	

]

8.46 Header: ara/diag/event.h

8.46.1 Class: Event

[SWS_DM_00646] Definition of API class ara::diag::Event

Upstream requirements: [RS_Diag_04151](#)

[

Kind:	class
Port Interfaces:	DiagnosticEventInterface
Header file:	#include "ara/diag/event.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	Event
Syntax:	<code>class Event final {...};</code>
Description:	Class to implement operations on diagnostic Events.

]

8.46.1.1 Public Member Functions

8.46.1.1.1 Special Member Functions

8.46.1.1.1.1 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:	<code>Event (Event &&) noexcept=delete;</code>
Description:	Move constructor of Event.

]

8.46.1.1.1.2 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:	<code>class ara::diag::Event</code>
Syntax:	<code>Event & operator= (Event &&) = delete;</code>
Description:	Move assignment operator of Event.

8.46.1.1.1.3 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:	<code>class ara::diag::Event</code>
Syntax:	<code>~Event () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class Event.

8.46.1.1.2 Constructors

8.46.1.1.2.1 Event(const ara::core::InstanceSpecifier&)

[SWS_DM_00647] Definition of API function ara::diag::Event::Event

Upstream requirements: [RS_Diag_04151](#), [RS_AP_00137](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function
Header file:	#include "ara/diag/event.h"
Scope:	<code>class ara::diag::Event</code>





Syntax:	explicit Event (const ara::core::InstanceSpecifier &specifier) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticEventInterface .
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.
	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.
	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.
Description:	Constructor Event class.	

]

8.46.1.1.2.2 Event(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.

]

8.46.1.1.3 Member Functions**8.46.1.1.3.1 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:	<code>class ara::diag::Event</code>	
Syntax:	<code>ara::core::Future< std::uint32_t > GetDTCNumber (DTCFormatType dtcFormat) noexcept;</code>	
Parameters (in):	dtcFormat	Define the DTC format for the return value.
Return value:	<code>ara::core::Future< std::uint32_t ></code>	Return the DTC number in respective DTC format .
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kNoSuchDTC	rollback_semantics
		No mapped DTC exits for the requested event.
	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the DTC number related to this event instance.	

]

8.46.1.1.3.2 GetDebouncingStatus**[SWS_DM_00654] Definition of API function ara::diag::Event::GetDebouncingStatus**Upstream requirements: [RS_Diag_04068](#), [RS_Diag_04225](#), [RS_AP_00139](#)

[

Kind:	function	
Header file:	#include "ara/diag/event.h"	
Scope:	<code>class ara::diag::Event</code>	
Syntax:	<code>ara::core::Future< DebouncingState > GetDebouncingStatus () noexcept;</code>	
Return value:	<code>ara::core::Future< DebouncingState ></code>	Return the current debouncing state of this event.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Get the current debouncing status.	

]

8.46.1.1.3.3 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 >	Return the current diagnostic event status
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current diagnostic event status.	

]

8.46.1.1.3.4 GetFaultDetectionCounter

[SWS_DM_00656] Definition of API function ara::diag::Event::GetFaultDetectionCounter

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 >	Returns the current FaultDetectionCounter value.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Returns the current value of the Fault Detection Counter (FDC) of this event.	

]

8.46.1.1.3.5 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:	DiagErrc::kServiceNot Available	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).	

8.46.1.1.3.6 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 (EventStatus ByteNotifier notifier) noexcept;	
Parameters (in):	notifier	The function to be called if a diagnostic event is changed.
Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

8.46.1.1.3.7 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:	<code>class ara::diag::Event</code>
Syntax:	<code>Event & operator= (Event &)=delete;</code>
Description:	Event shall be a single not assign-able instance.

]

8.47 Header: ara/diag/event_types.h

8.47.1 Non-Member Types

8.47.1.1 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 [9] specification.
	kDTCFormatUDS	= 1 ISO_14229-1_DTCFormat as defined in ISO 14229-1 [30] specification.
	kDTCFormatJ1939	= 2 SAE_J1939-73_DTCFormat as defined in SAE J1939-73 [31] specification.
Description:	Represents the type of the DTC format according to ISO 14229-1 [30] .	

]

8.47.1.2 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.

8.47.1.3 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
	kTestFailedThisOperationCycle	bit 1: TestFailedThisOperationCycle
	kTestNotCompletedThisOperationCycle	bit 6: TestNotCompletedThisOperationCycle
]		





Description:	Single event status bits.

]

8.47.1.4 Type Alias: EventStatusByteNotifier

[SWS_DM_02118] Definition of API type `ara::diag::EventStatusByteNotifier`

Upstream requirements: [RS_Diag_04183](#), [RS_AP_00139](#)

[

Kind:	type alias
Header file:	#include "ara/diag/event_types.h"
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>EventStatusByteNotifier</code>
Syntax:	<code>using EventStatusByteNotifier = std::function<void(EventStatusByte)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if a diagnostic event is changed.

]

8.47.2 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 <code>ara::diag</code>
Symbol:	<code>EventStatusByte</code>
Syntax:	<code>struct EventStatusByte { ... };</code>
Description:	Current event status byte, bit-encoded.

]

8.47.2.1 Public Member Functions

8.47.2.1.1 Special Member Functions

8.47.2.1.1.1 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 &other) noexcept=default;	
Parameters (in):	other	Object to copy-construct from.
Exception Safety:	exception safe	
Description:	Copy constructor.	

]

8.47.2.1.1.2 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 &&other) noexcept;	
Parameters (in):	other	Object to move-construct from.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructor.	

]

8.47.2.1.1.3 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 &other) noexcept;	
Parameters (in):	other	Object to copy-assign from.
Return value:	EventStatusByte &	EventStatusByte & Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator.	

]

8.47.2.1.1.4 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 &&other) noexcept;	
Parameters (in):	other	Object to move-assign from.
Return value:	EventStatusByte &	EventStatusByte & Reference to self.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator.	

]

8.47.2.1.2 Constructors

8.47.2.1.2.1 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:	<pre>template <typename... Args> constexpr EventStatusByte (Args... bits) noexcept;</pre>	
Parameters (in):	bits	The EventStatusBits that shall be set.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct an EventStatusByte where all EventStatusBits passed as parameter are set.	

]

8.47.2.1.3 Member Functions

8.47.2.1.3.1 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:	<pre>constexpr bool IsFailedAndTested () const noexcept;</pre>	
Return value:	bool	TRUE: if bits kTestFailed AND kTestFailedThisOperationCycle are set, else FALSE
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Retrieve the value of the bits kTestFailed and kTestFailedThisOperationCycle .	

]

8.47.2.1.3.2 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;	
Parameters (in):	bits	The EventStatusBits to check.
Return value:	bool	TRUE: if all EventStatusBits passed as parameter are NOT set, else FALSE.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether all EventStatusBits passed as parameter are not set.	

]

8.47.2.1.3.3 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	TRUE: if bits kTestFailed AND kTestNotCompletedThisOperationCycle are NOT set, else FALSE
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Retrieve the value of the bits kTestFailed and kTestNotCompletedThisOperationCycle .	

]

8.47.2.1.3.4 IsSet

[SWS_DM_01755] Definition of API function ara::diag::EventStatusByte::IsSetUpstream requirements: [RS_Diag_04151](#)

[

Kind:	function	
Header file:	#include "ara/diag/event_types.h"	
Scope:	struct ara::diag::EventStatusByte	
Syntax:	<pre>template <typename... Args> constexpr bool IsSet (Args... bits) const noexcept;</pre>	
Parameters (in):	bits	The EventStatusBits to check.
Return value:	bool	TRUE: if all EventStatusBits passed as parameter are set, else FALSE.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Whether all EventStatusBits passed as parameter are set.	

]

8.48 Header: ara/diag/dtc_information.h

8.48.1 Non-Member Types

8.48.1.1 Enumeration: ControlDtcStatusType

[SWS_DM_00663] Definition of API enum ara::diag::ControlDtcStatusTypeUpstream 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.	

]

8.48.1.2 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	
	kTestFailedThisOperationCycle	= 0x02 bit 1: TestFailedThisOperationCycle	
	kPendingDTC	= 0x04 bit 2: PendingDTC	
	kConfirmedDTC	= 0x08 bit 3: ConfirmedDTC	
	kTestNotCompletedSinceLastClear	= 0x10 bit 4: TestNotCompletedSinceLastClear	
	kTestFailedSinceLastClear	= 0x20 bit 5: TestFailedSinceLastClear	
	kTestNotCompletedThisOperationCycle	= 0x40 bit 6: TestNotCompletedThisOperationCycle	
	kWarningIndicatorRequested	= 0x80 bit 7: WarningIndicatorRequested	
	Description:	UDS DTC status bits according to ISO 14229-1.	

8.48.2 Class: DTCInformation

[SWS_DM_00657] Definition of API class ara::diag::DTCInformation

Upstream requirements: [RS_Diag_04150](#), [RS_Diag_04105](#)

[	
Kind:	class
Port Interfaces:	DiagnosticDTCInformationInterface
Header file:	#include "ara/diag/dtc_information.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DTCInformation
]	





Syntax:	<code>class DTCInformation final {...};</code>
Description:	Class to implement operations on DTC informations per configured DiagnosticMemory Destination.

]

8.48.2.1 Public Member Types

8.48.2.1.1 Type Alias: ControlDtcStatusTypeNotifier

[SWS_DM_02125] Definition of API type `ara::diag::DTCInformation::ControlDtcStatusTypeNotifier`

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04159](#)

[

Kind:	type alias
Header file:	<code>#include "ara/diag/dtc_information.h"</code>
Scope:	<code>class ara::diag::DTCInformation</code>
Symbol:	<code>ControlDtcStatusTypeNotifier</code>
Syntax:	<code>using ControlDtcStatusTypeNotifier = std::function<void(ControlDtcStatusType)>;</code>
Thread Safety:	not thread-safe
Description:	Registers a notifier function which is called if the control DTC setting is changed.

]

8.48.2.1.2 Type Alias: DtcStatusChangedNotifier

[SWS_DM_02122] Definition of API type `ara::diag::DTCInformation::DtcStatusChangedNotifier`

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04148](#)

[

Kind:	type alias
Header file:	<code>#include "ara/diag/dtc_information.h"</code>
Scope:	<code>class ara::diag::DTCInformation</code>
Symbol:	<code>DtcStatusChangedNotifier</code>
Syntax:	<code>using DtcStatusChangedNotifier = std::function<void(std::uint32_t dtc, UdsDtcStatusByteType udsStatusByteOld, UdsDtcStatusByteType udsStatusByteNew)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if a UDS DTC status is changed.

]

8.48.2.1.3 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:	<code>class ara::diag::DTCInformation</code>
Symbol:	EventMemoryOverflowSetNotifier
Syntax:	<code>using EventMemoryOverflowSetNotifier = std::function<void(bool)>;</code>
Thread Safety:	not thread-safe
Description:	Notifier function which is called if the current event memory overflow status changed.

]

8.48.2.1.4 Type Alias: NumberOfStoredEntriesNotifier

[SWS_DM_02124] Definition of API type ara::diag::DTCInformation::NumberOfStoredEntriesNotifier

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04109](#)

[

Kind:	type alias
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>class ara::diag::DTCInformation</code>
Symbol:	NumberOfStoredEntriesNotifier
Syntax:	<code>using NumberOfStoredEntriesNotifier = std::function<void(std::uint32_t)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the number of currently stored fault memory entries changed.

]

8.48.2.1.5 Type Alias: SnapshotRecordUpdatedTypeNotifier

[SWS_DM_02123] Definition of API type `ara::diag::DTCInformation::SnapshotRecordUpdatedTypeNotifier`

Upstream requirements: [RS_AP_00139](#), [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	type alias
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>class ara::diag::DTCInformation</code>
Symbol:	<code>SnapshotRecordUpdatedTypeNotifier</code>
Syntax:	<code>using SnapshotRecordUpdatedTypeNotifier = std::function<void(SnapshotRecordUpdatedType)>;</code>
Thread Safety:	not thread-safe
Description:	Register a notifier function which is called if the SnapshotRecord is changed.

]

8.48.2.2 Public Member Functions

8.48.2.2.1 Special Member Functions

8.48.2.2.1.1 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:	<code>class ara::diag::DTCInformation</code>
Syntax:	<code>DTCInformation (DTCInformation &&) noexcept=delete;</code>
Description:	Move constructor of DTCInformation.

]

8.48.2.2.1.2 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:	<code>class ara::diag::DTCInformation</code>
Syntax:	<code>DTCInformation & operator= (DTCInformation &&)=delete;</code>
Description:	Move assignment operator of DTCInformation.

]

8.48.2.2.1.3 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:	<code>class ara::diag::DTCInformation</code>
Syntax:	<code>~DTCInformation () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class DTCInformation.

]

8.48.2.2.2 Constructors

8.48.2.2.2.1 DTCInformation(const ara::core::InstanceSpecifier&)

[SWS_DM_00664] Definition of API function `ara::diag::DTCInformation::DTCInformation`

Upstream requirements: [RS_AP_00137](#), [RS_Diag_04150](#), [RS_Diag_04105](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>class ara::diag::DTCInformation</code>





Syntax:	explicit DTCInformation (const ara::core::InstanceSpecifier &specifier) noexcept;	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticDTCInformationInterface .
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.
	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.
	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.
Description:	Constructor for a DTCInformation instance which allows for DTC related operation per DiagnosticMemoryDestination.	

]

8.48.2.2.2.2 DTCInformation(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.

]

8.48.2.2.3 Member Functions

8.48.2.2.3.1 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Future< void > Clear (std::uint32_t dtcGroup) noexcept;</code>	
Parameters (in):	dtcGroup	DTC group to be cleared.
Return value:	<code>ara::core::Future< void ></code>	void or errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kBusy	rollback_semantics
		Interface is busy with processing.
	DiagErrc::kFailed	rollback_semantics
		Failed to process.
	DiagErrc::kMemoryError	rollback_semantics
		A memory error occurred during processing.
	DiagErrc::kWrongDtc	rollback_semantics
		A wrong DTC number was requested.
	DiagErrc::kServiceNot Available	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.	

]

8.48.2.2.3.2 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Result< void > EnableControlDtc () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽



Errors:	This function does not specify any standardized errors.
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.

]

8.48.2.2.3.3 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Future< ControlDtcStatusType > GetControlDTCStatus () noexcept;</code>	
Return value:	<code>ara::core::Future< ControlDtcStatusType ></code>	The current status of ControlDtcStatus (related to UDS service 0x85)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Contains the current status of the ControlDTCStatus.	

]

8.48.2.2.3.4 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Future< UdsDtcStatusByteType > GetCurrentStatus (std::uint32_t dtc) noexcept;</code>	
Parameters (in):	dtc	DTC identifier for which the status should be retrieved.





Return value:	ara::core::Future< Uds DtcStatusByteType >	A Future which is readied with current UDS DTC status byte of the given DTC identifier or an internal error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kNoSuchDTC	rollback_semantics
		No mapped DTC exists for the requested event.
	DiagErrc::kServiceNot Available	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.	

]

8.48.2.2.3.5 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:	DiagErrc::kServiceNot Available	rollback_semantics
		The call cannot be executed, because essential DM functionality is currently not available.
Description:	Contains the current event memory overflow status.	

]

8.48.2.2.3.6 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Future< std::uint32_t > GetNumberOfStoredEntries () noexcept;</code>	
Return value:	<code>ara::core::Future< std::uint32_t ></code>	Number of currently stored fault memory entries.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	<code>DiagErrc::kServiceNot Available</code>	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.	

8.48.2.2.3.7 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Result< void > SetControlDtcStatusNotifier (ControlDtcStatusTypeNotifier notifier) noexcept;</code>	
Parameters (in):	<code>notifier</code>	The function to be called if the ControlDTCStatus (related to UDS service 0x85) for this diagnostic memory instance has changed.
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagErrc::kInvalid Argument</code>	rollback_semantics e.g. kPreFailed with internal debouncing
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.	

8.48.2.2.3.8 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 (DtcStatus ChangedNotifier notifier) noexcept;	
Parameters (in):	notifier	The function to be called if a DTC status has changed.
Return value:	ara::core::Result< void >	Returns void.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	This function does not specify any standardized errors.	
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.	

]

8.48.2.2.3.9 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 (EventMemory OverflowSetNotifier 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics e.g. kPreFailed with internal debouncing
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.	

]

8.48.2.2.3.10 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Result< void > SetNumberOfStoredEntriesNotifier (NumberOfStoredEntriesNotifier notifier) noexcept;</code>	
Parameters (in):	notifier	The function to be called if the number of entries for this diagnostic event memory instance has changed.
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
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.	

8.48.2.2.3.11 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:	<code>class ara::diag::DTCInformation</code>	
Syntax:	<code>ara::core::Result< void > SetSnapshotRecordUpdatedNotifier (SnapshotRecordUpdatedTypeNotifier notifier) noexcept;</code>	
Parameters (in):	notifier	The function to be called if the SnapshotRecord is changed.
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing





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.

]

8.48.2.2.3.12 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.

]

8.48.3 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 DiagnosticDataIdentifier status.

]

8.48.3.1 Public Member Variables

8.48.3.1.1 data

[SWS_DM_02113] Definition of API variable `ara::diag::DTCInformation::SnapshotDataIdentifierType::data`

Upstream requirements: [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/dtc_information.h"</code>
Scope:	<code>struct ara::diag::DTCInformation::SnapshotDataIdentifierType</code>
Symbol:	<code>data</code>
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> data;</code>
Description:	Data for specific snapshot record .

]

8.48.3.1.2 dataIdentifier

[SWS_DM_02112] Definition of API variable `ara::diag::DTCInformation::SnapshotDataIdentifierType::dataIdentifier`

Upstream requirements: [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/dtc_information.h"</code>
Scope:	<code>struct ara::diag::DTCInformation::SnapshotDataIdentifierType</code>
Symbol:	<code>dataIdentifier</code>
Type:	<code>std::uint16_t</code>
Syntax:	<code>std::uint16_t dataIdentifier;</code>
Description:	The DiagnosticDataIdentifier identifying the snapshot record .

]

8.48.4 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:	<code>class ara::diag::DTCInformation</code>
Symbol:	SnapshotDataRecordType
Syntax:	<code>struct SnapshotDataRecordType {...};</code>
Description:	Type for <code>snapshot record</code> status.

]

8.48.4.1 Public Member Variables

8.48.4.1.1 snapshotDataIdentifiers

[SWS_DM_02115] Definition of API variable ara::diag::DTCInformation::SnapshotDataRecordType::snapshotDataIdentifiers

Upstream requirements: [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	variable
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>struct ara::diag::DTCInformation::SnapshotDataRecordType</code>
Symbol:	snapshotDataIdentifiers
Type:	<code>ara::core::Vector< SnapshotDataIdentifierType ></code>
Syntax:	<code>ara::core::Vector<SnapshotDataIdentifierType> snapshotDataIdentifiers;</code>
Description:	Vector of snapshot <code>DagnosticDataIdentifier</code> s.

]

8.48.4.1.2 snapshotRecordNumber

[SWS_DM_02114] Definition of API variable ara::diag::DTCInformation::SnapshotDataRecordType::snapshotRecordNumber

Upstream requirements: [RS_Diag_04205](#)

[

Kind:	variable
Header file:	#include "ara/diag/dtc_information.h"
Scope:	struct ara::diag::DTCInformation::SnapshotDataRecordType
Symbol:	snapshotRecordNumber
Type:	std::uint8_t
Syntax:	std::uint8_t snapshotRecordNumber;
Description:	Specifies a unique record number for a snapshot record .

]

8.48.5 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 snapshot record updated status.

]

8.48.5.1 Public Member Variables

8.48.5.1.1 DTC

[SWS_DM_02116] Definition of API variable `ara::diag::DTCInformation::SnapshotRecordUpdatedType::DTC`

Upstream requirements: [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	variable
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>struct ara::diag::DTCInformation::SnapshotRecordUpdatedType</code>
Symbol:	DTC
Type:	uint32_t
Syntax:	<code>uint32_t DTC;</code>
Description:	DTC identifier to which the <code>snapshot record</code> data is related.

]

8.48.5.1.2 snapshotDataRecords

[SWS_DM_02117] Definition of API variable `ara::diag::DTCInformation::SnapshotRecordUpdatedType::snapshotDataRecords`

Upstream requirements: [RS_Diag_04205](#), [RS_Diag_04091](#)

[

Kind:	variable
Header file:	#include "ara/diag/dtc_information.h"
Scope:	<code>struct ara::diag::DTCInformation::SnapshotRecordUpdatedType</code>
Symbol:	snapshotDataRecords
Type:	<code>ara::core::Vector< SnapshotDataRecordType ></code>
Syntax:	<code>ara::core::Vector<SnapshotDataRecordType> snapshotDataRecords;</code>
Description:	Vector of snapshot <code>snapshot records</code> .

]

8.48.6 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.

]

8.48.6.1 Public Member Functions

8.48.6.1.1 Special Member Functions

8.48.6.1.1.1 Move Constructor

[SWS_DM_02086] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType

Upstream requirements: [RS_Diag_04067](#)

[

Kind:	function
Header file:	#include "ara/diag/dtc_information.h"
Scope:	struct ara::diag::DTCInformation::UdsDtcStatusByteType
Syntax:	constexpr UdsDtcStatusByteType (UdsDtcStatusByteType &&other) noexcept;
Parameters (in):	other Object to move-construct from.
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Move constructor.

]

8.48.6.1.1.2 Copy Constructor**[SWS_DM_02085] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType***Upstream requirements:* [RS_Diag_04067](#)

[

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	struct ara::diag::DTCInformation::UdsDtcStatusByteType	
Syntax:	constexpr UdsDtcStatusByteType (const UdsDtcStatusByteType &other) noexcept=default;	
Parameters (in):	other	Object to copy-construct from.
Exception Safety:	exception safe	
Description:	Copy constructor.	

]

8.48.6.1.1.3 Copy Assignment Operator**[SWS_DM_02087] Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::operator=***Upstream requirements:* [RS_Diag_04067](#)

[

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	struct ara::diag::DTCInformation::UdsDtcStatusByteType	
Syntax:	constexpr UdsDtcStatusByteType & operator= (const UdsDtcStatusByteType &other) noexcept;	
Parameters (in):	other	Object to copy-assign from
Return value:	UdsDtcStatusByteType &	UdsDtcStatusByteType & Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator.	

]

8.48.6.1.1.4 Move Assignment Operator

[SWS_DM_02088] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::operator=`

Upstream requirements: [RS_Diag_04067](#)

[

Kind:	function	
Header file:	#include "ara/diag/dtc_information.h"	
Scope:	struct ara::diag::DTCInformation::UdsDtcStatusByteType	
Syntax:	constexpr UdsDtcStatusByteType & operator= (UdsDtcStatusByteType &&other) noexcept;	
Parameters (in):	other	Object to move-assign from
Return value:	UdsDtcStatusByteType &	UdsDtcStatusByteType& Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator.	

]

8.48.6.1.2 Constructors

8.48.6.1.2.1 UdsDtcStatusByteType

[SWS_DM_02084] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType`

Upstream requirements: [RS_Diag_04067](#)

[

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;	
Parameters (in):	bits	The DTC bits to be set.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Construct an UdsDtcStatusByteType where all DtcStatusBits passed as parameter are set.	

]

8.48.6.1.3 Member Functions

8.48.6.1.3.1 IsFailedAndTested

[SWS_DM_02080] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::IsFailedAndTested`

Upstream requirements: [RS_Diag_04067](#)

[

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	bool Returns <code>TRUE</code> if DTC is Failed AND Tested, else <code>FALSE</code> .
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Test if DTC is Failed and Tested.	

]

8.48.6.1.3.2 IsNotSet

[SWS_DM_02083] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::IsNotSet`

Upstream requirements: [RS_Diag_04067](#)

[

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;	
Parameters (in):	bits	The DTC bits to check
Return value:	bool	bool Returns <code>TRUE</code> if DTC status bit(s) is not set, else <code>FALSE</code> .
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Test if DTC status bit(s) is not set.	

]

8.48.6.1.3.3 IsPassedAndTested

[SWS_DM_02081] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::IsPassedAndTested`

Upstream requirements: [RS_Diag_04067](#)

[

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	bool Returns <code>TRUE</code> if DTC is Passed AND Tested, else <code>FALSE</code> .
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Test if DTC is Passed and Tested.	

]

8.48.6.1.3.4 IsSet

[SWS_DM_02082] Definition of API function `ara::diag::DTCInformation::UdsDtcStatusByteType::IsSet`

Upstream requirements: [RS_Diag_04067](#)

[

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;	
Parameters (in):	bits	The DTC bits to check
Return value:	bool	bool Returns <code>TRUE</code> if DTC status bit(s) is set, else <code>FALSE</code> .
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Test if DTC status bit(s) is set.	

]

8.49 Header: {<extended-data-record-element-shortname-lower>}.h

[SWS_DM_02157] Diagnostic ExtendedDataRecordElement Header File: file name, includes and multiple inclusion guard

Status: DRAFT

Upstream requirements: [RS_AP_00114](#)

[

Kind:	Header File	
Syntax:	{<extended-data-record-element-shortname-lower>}.h/ {<extended-data-record-element-shortname-lower>}.h	
Description:	For each modeled DiagnosticExtendedDataRecordInterface a Diagnostic ExtendedDataRecordElement 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:	{<extended-data-record-element-shortname-lower>}	as per DiagnosticExtendedDataRecordInterface . shortName converted to lower-case.
Example:	<pre>// File=n/n_plus_1/n_plus_2/si_extended-data-record-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)</pre>	

]

8.49.1 Namespaces

8.49.1.1 {<namespace-list-extended-data-record-element>}

[SWS_DM_02159] Diagnostic ExtendedDataRecordElement Header File: service namespace

Status: DRAFT

Upstream requirements: [RS_AP_00114](#)

[

Kind:	namespace	
Header file:	#include "{<extended-data-record-element-shortname-lower>}.h"	
Scope:	--	
Syntax:	namespace {<namespace-list-extended-data-record-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-extended-data-record-element>}	as per namespace in the ordered list: namespace [N+1] shall be an inner namespace of namespace [N] converted to lower-case.

]

8.49.2 Class: {<extended-data-record-element-interface-name-x>}

[SWS_DM_02146] Definition of API class {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}

Status: DRAFT

Upstream requirements: RS_Diag_04169

[

Kind:	class	
Header file:	#include "{<extended-data-record-element-shortname-lower>}.h"	
Forwarding header file:	#include "{<extended-data-record-element-shortname-lower>}/_fwd.h"	
Scope:	namespace {<namespace-list-extended-data-record-element>}	
Symbol:	{<extended-data-record-element-interface-name-x>}	
Syntax:	class {<extended-data-record-element-interface-name-x>} {...};	
Description:	DiagnosticExtendedDataRecordInterface class	
Descriptors:	{<extended-data-record-element-interface-name-x>}	The <code>DiagnosticExtendedDataRecordInterface</code> . <code>shortName</code> converted to upper camel-case letters

]

8.49.2.1 Public Member Variables**8.49.2.1.1 {<extended-data-record-element-out-arg-symbol>}**

[SWS_DM_02155] Definition of API variable {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::{<extended-data-record-element-out-arg-symbol>}

Status: DRAFT

Upstream requirements: RS_Diag_04224, RS_Diag_04169

[

Kind:	variable	
Header file:	#include "{<extended-data-record-element-shortname-lower>}.h"	
Scope:	class {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}	
Symbol:	{<extended-data-record-element-out-arg-symbol>}	
Type:	{<extended-data-record-element-out-arg-type>}	
Syntax:	{<extended-data-record-element-out-arg-type>}{<extended-data-record-element-out-arg-symbol>;	
Description:	Member declaration representing an <i>out</i> argument in an <i>Output</i> .	
Descriptors:	{<extended-data-record-element-out-arg-type>}	The <code>ClientServerOperation.argument.type</code> , mapped to a C++ data type according to [27].

▽



	{<extended-data-record-element-out-arg-symbol>}	Symbol name of the struct element as given by ClientServerOperation. ArgumentDataPrototype.shortName

]

8.49.2.2 Public Member Functions

8.49.2.2.1 Member Functions

8.49.2.2.1.1 Offer

[SWS_DM_02152] Definition of API function {<namespace-list-extended-data-record-element>}:: {<extended-data-record-element-interface-name-x>}::Offer

Upstream requirements: RS_AP_00119, RS_AP_00139, RS_Diag_04169, RS_Diag_04224

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	class {<namespace-list-extended-data-record-element>}:: {<extended-data-record-element-interface-name-x>}	
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:	thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler	

]

8.49.2.2.1.2 Read

[SWS_DM_02156] Definition of API function {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::Read

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	class {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}	
Syntax:	virtual ara::core::Future< <extended-data-record-element-name-upper-camel>ReadOutput > Read (const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept=0;	
Parameters (in):	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter is used to - Query the current cancellation status by calling <code>cancellationHandler.IsCanceled</code>. Returns <code>TRUE</code> in case, the current conversation has been cancelled and <code>FALSE</code> otherwise. - Register a notifier function by calling <code>cancellationHandler.SetNotifier</code>. The registered notifier is called if the current conversation is cancelled.
Return value:	ara::core::Future< { <extended-data-record-element-name-upper-camel> >ReadOutput >	- If successful: an <code>ara::core::Future</code> containing an <code>ReadOutput</code> object as per [SWS_DM_02154] - If unsuccessful: an <code>ara::core::Future</code> containing a corresponding <code>DiagnosticExtendedDataRecordInterface</code> or <code>ApApplicationError</code>.
Exception Safety:	exception safe	
Thread Safety:	conditional	Thread safety is implementation defined
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
Description:	Provision of a Read ExtendedDataRecordElement. The data fetched by this method is stored in non-volatile memory by the DM and therefore always a value needs to be returned. Not available data at the time of the method call shall be filled with a filling pattern, e.g 0xFF by the implementation of this method (<code>DiagnosticExtendedDataRecordInterface</code> in the role <code>provide</code>).	
Descriptors:	<extended-data-record-element-name-upper-camel>	Name of a DataElement created from the <code>ClientServerOperation.shortName</code> defined in the <code>DiagnosticExtendedDataRecordInterface</code> in the role <code>provide</code> converted to upper camel-case letters.
Example:	<pre>// Example: virtual ara::core::Future<SomeDataElementReadOutput> Read(const ara::diag::MetaInfo& metaInfo, ara::diag::CancellationHandler cancellationHandler) noexcept = 0;</pre>	

8.49.2.2.1.3 StopOffer

[SWS_DM_02153] Definition of API function {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::StopOffer

Upstream requirements: [RS_Diag_04169](#)

[

Kind:	function
Header file:	#include "<extended-data-record-element-shortname-lower>.h"
Scope:	class {<namespace-list-extended-data-record-element>}::{ <extended-data-record-element-interface-name-x>}
Syntax:	void StopOffer () noexcept;
Return value:	None
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM

]

8.49.2.2.1.4 operator=

[SWS_DM_02148] Definition of API function {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::operator=

Status: DRAFT

Upstream requirements: [RS_AP_00147](#)

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	class {<namespace-list-extended-data-record-element>}::{ <extended-data-record-element-interface-name-x>}	
Syntax:	{<extended-data-record-element-interface-name-3> & operator= (const <extended-data-record-element-interface-name-3> &other)=delete;	
Description:	Copy assignment constructor deletion	
Descriptors:	<extended-data-record-element-interface-name-3>	As per { <extended-data-record-element-interface-name-x> in [SWS_DM_02146]

]

8.49.2.2.1.5 operator=

[SWS_DM_02150] Definition of API function {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::operator=

Status: DRAFT

Upstream requirements: RS_AP_00147

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	class {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}	
Syntax:	{<extended-data-record-element-interface-name-5>} & operator= ({<extended-data-record-element-interface-name-5>} &&other)=delete;	
Description:	Move assignment constructor	
Descriptors:	{<extended-data-record-element-interface-name-5>}	As per {<extended-data-record-element-interface-name-x>} in [SWS_DM_02146]

]

8.49.2.2.1.6 {<extended-data-record-element-interface-name-1>}

[SWS_DM_02158] Definition of API function {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}::{<extended-data-record-element-interface-name-1>}

Upstream requirements: RS_AP_00137

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	class {<namespace-list-extended-data-record-element>}::{<extended-data-record-element-interface-name-x>}	
Syntax:	{<extended-data-record-element-interface-name-1>} (ara::core::InstanceSpecifier instanceSpec, ara::diag::ConcurrencyType concurrencyType) noexcept;	
Parameters (in):	instanceSpec	The specifier of a specific instance of a service.
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.
	PortInterfaceMappingViolation	A PortPrototype that is typed by a DiagnosticExtendedDataRecordInterface needs to be referenced by a DiagnosticDataPortMapping.





	ProcessMappingViolation	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.
	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.
Description:	Construct a DataElement from an <code>ara::core::InstanceSpecifier</code>	
Descriptors:	<code>{<extended-data-record-element-interface-name-1>}</code>	As per { <code><extended-data-record-element-interface-name-x></code> in [SWS_DM_02146]

]

8.49.2.2.1.7 {<extended-data-record-element-interface-name-2>}

[SWS_DM_02147] Definition of API function `{<namespace-list-extended-data-record-element>>::{<extended-data-record-element-interface-name-x>>::{<extended-data-record-element-interface-name-2>}}`

Upstream requirements: RS_AP_00147

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	<code>class {<namespace-list-extended-data-record-element>>::{<extended-data-record-element-interface-name-x>}</code>	
Syntax:	<code>{<extended-data-record-element-interface-name-2>}</code> (const <code>{<extended-data-record-element-interface-name-2>}</code> &other)=delete;	
Description:	Copy constructor deletion	
Descriptors:	<code>{<extended-data-record-element-interface-name-2>}</code>	As per { <code><extended-data-record-element-interface-name-x></code> in [SWS_DM_02146]

]

8.49.2.2.1.8 {<extended-data-record-element-interface-name-4>}

[SWS_DM_02149] Definition of API function `{<namespace-list-extended-data-record-element>::~{<extended-data-record-element-interface-name-x>}}::<extended-data-record-element-interface-name-4>`

Upstream requirements: RS_AP_00147

[

Kind:	function	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Scope:	<code>class {<namespace-list-extended-data-record-element>>::{<extended-data-record-element-interface-name-x>}</code>	





Syntax:	<code>{<extended-data-record-element-interface-name-4>} ({<extended-data-record-element-interface-name-4> &&other)=delete;</code>	
Description:	Move constructor	
Descriptors:	<code>{<extended-data-record-element-interface-name-4>}</code>	As per { <extended-data-record-element-interface-name-x> in [SWS_DM_02146]

]

8.49.2.2.1.9 ~{<extended-data-record-element-interface-name-6>}

[SWS_DM_02151] Definition of API function {<namespace-list-extended-data-record-element>>::{<extended-data-record-element-interface-name-x>}::~{<extended-data-record-element-interface-name-6>}

Status: DRAFT

Upstream requirements: RS_Diag_04169, RS_AP_00134

[

Kind:	function	
Header file:	#include "<{<extended-data-record-element-shortname-lower>}.h"	
Scope:	<code>class {<namespace-list-extended-data-record-element>}:: {<extended-data-record-element-interface-name-x>}</code>	
Syntax:	<code>virtual ~{<extended-data-record-element-interface-name-6>} () noexcept;</code>	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Destruction of a <code>DataElement</code>	
Descriptors:	<code>{<extended-data-record-element-interface-name-6>}</code>	As per { <extended-data-record-element-interface-name-x> in [SWS_DM_02146]

]

8.49.3 Struct: {<extended-data-record-element-name-upper-camel>}ReadOutput

[SWS_DM_02154] Definition of API class {<namespace-list-extended-data-record-element>}:: {<extended-data-record-element-interface-name-x>}:: {<extended-data-record-element-name-upper-camel>}ReadOutput

Status: DRAFT

Upstream requirements: RS_AP_00114, RS_AP_00127, RS_AP_00128, RS_AP_00138

Kind:	struct	
Header file:	#include "<extended-data-record-element-shortname-lower>.h"	
Forwarding header file:	#include "<extended-data-record-element-shortname-lower>/_fwd.h"	
Scope:	class {<namespace-list-extended-data-record-element>}:: {<extended-data-record-element-interface-name-x>}	
Symbol:	{<extended-data-record-element-name-upper-camel>}ReadOutput	
Syntax:	struct {<extended-data-record-element-name-upper-camel>}ReadOutput { ... };	
Description:	Structure wrapping the <i>out</i> arguments for a ReadOutput	
Descriptors:	{<extended-data-record-element-name-upper-camel>}	As per {<extended-data-record-element-name-upper-camel>} in [SWS_DM_02154]
	{<extended-data-record-element-out-args-members-ordered>}	Shown as "..." in Syntax. Each element is an <i>argument</i> in the ordered list of <i>arguments</i> , with <i>direction</i> == <i>out</i> or <i>direction</i> == <i>inout</i> , where [SWS_DM_02158] applies for each.

8.50 Diagnostic SOVD-related APIs**8.51 Header: ara/diag/diag_sovd_error_domain.h****8.51.1 Non-Member Types****8.51.1.1 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, e.g. 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).
	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.
	kPreconditionNotFulfilled	= 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.	

]

8.51.2 Non-Member Functions

8.51.2.1 Other

8.51.2.1.1 GetDiagSovdErrorDomain

[SWS_DM_01816] Definition of API function ara::diag::GetDiagSovdErrorDomain

Upstream requirements: [RS_AP_00119](#)

[

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.

]

8.51.2.1.2 MakeErrorCode

[SWS_DM_01817] Definition of API function ara::diag::MakeErrorCode

Upstream requirements: [RS_AP_00119](#)

[

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.	

]

8.51.3 Class: DiagSovdErrorDomain

[SWS_DM_01809] Definition of API class ara::diag::DiagSovdErrorDomain

Upstream requirements: [RS_AP_00119](#)

[

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.

]

8.51.3.1 Public Member Types

8.51.3.1.1 Type Alias: Errc

[SWS_DM_01810] Definition of API type `ara::diag::DiagSovdErrorDomain::Errc`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	<code>class ara::diag::DiagSovdErrorDomain</code>
Symbol:	Errc
Syntax:	<code>using Errc = DiagSovdErrc;</code>
Description:	Alias for the error code value enumeration.

]

8.51.3.1.2 Type Alias: Exception

[SWS_DM_01811] Definition of API type `ara::diag::DiagSovdErrorDomain::Exception`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	type alias
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	<code>class ara::diag::DiagSovdErrorDomain</code>
Symbol:	Exception
Syntax:	<code>using Exception = DiagSovdException;</code>
Description:	Alias for the exception base class.

]

8.51.3.2 Public Member Functions

8.51.3.2.1 Special Member Functions

8.51.3.2.1.1 Default Constructor

[SWS_DM_01812] Definition of API function `ara::diag::DiagSovdErrorDomain::DiagSovdErrorDomain`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	<code>class ara::diag::DiagSovdErrorDomain</code>
Syntax:	<code>constexpr DiagSovdErrorDomain () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Default constructor.

]

8.51.3.2.2 Member Functions

8.51.3.2.2.1 Message

[SWS_DM_01814] Definition of API function `ara::diag::DiagSovdErrorDomain::Message`

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function	
Header file:	#include "ara/diag/diag_sovd_error_domain.h"	
Scope:	<code>class ara::diag::DiagSovdErrorDomain</code>	
Syntax:	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>	
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.	

]

8.51.3.2.2 Name

[SWS_DM_01813] Definition of API function `ara::diag::DiagSovdErrorDomain::Name`

Upstream requirements: [RS_AP_00119](#)

[

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 *	As per ara::diag::DiagSovdErrorDomain in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]

8.51.3.2.3 ThrowAsException

[SWS_DM_01815] Definition of API function `ara::diag::DiagSovdErrorDomain::ThrowAsException`

Upstream requirements: [RS_AP_00119](#)

[

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 <code>ErrorCode</code> instance
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throw the exception type corresponding to the given <code>ErrorCode</code> . As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

]

8.51.4 Class: DiagSovdException

[SWS_DM_01807] Definition of API class ara::diag::DiagSovdException

Upstream requirements: [RS_AP_00119](#)

[

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:	<code>class DiagSovdException : public ara::core::Exception {...};</code>
Description:	Exception type for SOVD thrown by Diag classes.

]

8.51.4.1 Public Member Functions

8.51.4.1.1 Constructors

8.51.4.1.1.1 DiagSovdException

[SWS_DM_01808] Definition of API function ara::diag::DiagSovdException::DiagSovdException

Upstream requirements: [RS_AP_00119](#)

[

Kind:	function
Header file:	#include "ara/diag/diag_sovd_error_domain.h"
Scope:	<code>class ara::diag::DiagSovdException</code>
Syntax:	<code>explicit DiagSovdException (ara::core::ErrorCode err) noexcept;</code>
Parameters (in):	err the ErrorCode
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Construct a new DiagSovdException from an ErrorCode.

]

8.52 Header: ara/diag/sovd_authorization.h

8.52.1 Class: SovdAuthorization

[SWS_DM_01490] Definition of API class ara::diag::SovdAuthorization

Upstream requirements: [RS_Diag_04268](#)

[

Kind:	class
Port Interfaces:	DiagnosticSovdAuthorizationInterface
Header file:	#include "ara/diag/sovd_authorization.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	SovdAuthorization
Syntax:	<code>class SovdAuthorization {...};</code>
Description:	Class to implement the SOVD Authorization interfaces in the application.

]

8.52.1.1 Public Member Functions

8.52.1.1.1 Special Member Functions

8.52.1.1.1.1 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:	<code>virtual ~SovdAuthorization () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdAuthorization.

]

8.52.1.1.2 Constructors

8.52.1.1.2.1 SovdAuthorization

[SWS_DM_01489] Definition of API function `ara::diag::SovdAuthorization::SovdAuthorization`

Upstream requirements: [RS_Diag_04268](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/sovd_authorization.h"	
Scope:	<code>class ara::diag::SovdAuthorization</code>	
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 (kNotConcurrent).
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.
	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.
	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.
Description:	Constructs the port for SOVD Authorization interface.	

8.52.1.1.3 Member Functions

8.52.1.1.3.1 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:	<code>class ara::diag::SovdAuthorization</code>





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 >	Returns 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
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.	

]

8.52.1.1.3.2 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 >	Returns 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
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.	

]

8.52.1.1.3.3 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:	<code>class ara::diag::SovdAuthorization</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics The service is already offered.
	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics monitor configuration does not match DEXT
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.52.1.1.3.4 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:	<code>class ara::diag::SovdAuthorization</code>	
Syntax:	<code>void StopOffer () noexcept;</code>	
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

]

8.52.1.1.3.5 ValidateAuthorization

[SWS_DM_01483] Definition of API function ara::diag::SovdAuthorization::ValidateAuthorizationUpstream 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< ValidateAuthorizationOutput >	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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagSovdErrc::kInvalidAccessToken	rollback_semantics
		The access token provided by the client is invalid.
	DiagSovdErrc::kTemporarilyNotAvailable	rollback_semantics The request currently can't be served, e.g. 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 SOVD 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.	

]

8.52.2 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:	<code>class ara::diag::SovdAuthorization</code>
Symbol:	ValidateAuthorizationOutput
Syntax:	<code>struct ValidateAuthorizationOutput {...};</code>
Description:	Return type for ValidateAuthorization().

]

8.52.2.1 Public Member Variables

8.52.2.1.1 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:	<code>struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput</code>
Symbol:	identity
Type:	<code>ara::core::Optional< ara::core::String ></code>
Syntax:	<code>ara::core::Optional<ara::core::String> identity;</code>
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.

]

8.52.2.1.2 validUntil

[SWS_DM_01820] Definition of API variable `ara::diag::SovdAuthorization::ValidateAuthorizationOutput::validUntil`

Upstream requirements: [RS_Diag_04268](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/sovd_authorization.h"</code>
Scope:	<code>struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput</code>
Symbol:	<code>validUntil</code>
Type:	<code>std::chrono::system_clock::time_point</code>
Syntax:	<code>std::chrono::system_clock::time_point validUntil;</code>
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).

]

8.53 Header: `ara/diag/sovd_bulk_data.h`

8.53.1 Class: `SovdBulkData`

[SWS_DM_01841] Definition of API class `ara::diag::SovdBulkData`

Upstream requirements: [RS_Diag_04266](#)

[

Kind:	class
Port Interfaces:	DiagnosticSovdBulkDataInterface
Header file:	<code>#include "ara/diag/sovd_bulk_data.h"</code>
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>SovdBulkData</code>
Syntax:	<code>class SovdBulkData {...};</code>
Description:	Generic SOVD Bulk Data interface.

]

8.53.1.1 Public Member Functions

8.53.1.1.1 Special Member Functions

8.53.1.1.1.1 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:	<code>class ara::diag::SovdBulkData</code>
Syntax:	<code>virtual ~SovdBulkData () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdBulkData.

]

8.53.1.1.2 Constructors

8.53.1.1.2.1 SovdBulkData

[SWS_DM_01847] Definition of API function `ara::diag::SovdBulkData::SovdBulkData`

Upstream requirements: [RS_Diag_04266](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/sovd_bulk_data.h"	
Scope:	<code>class ara::diag::SovdBulkData</code>	
Syntax:	<code>SovdBulkData (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
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 (kNotConcurrent).
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.





	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.
	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.
Description:	Constructor for the SovdBulkData Interface.	

]

8.53.1.1.3 Member Functions

8.53.1.1.3.1 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
Return value:	ara::core::Future< DeleteByCategoryResult >	A Future with a "DeleteByCategoryResult" or a SOVD error code.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for SOVD method "Delete All Bulk Data Defined by Category".	

]

8.53.1.1.3.2 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:	<code>class ara::diag::SovdBulkData</code>	
Syntax:	<pre>virtual ara::core::Future< void > DeleteSpecificBulkData (ara::core::String bulkDataCategory, ara::core::String bulkDataId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	bulkDataCategory	Bulk data category of the bulk data instance to be deleted
	bulkDataId	Bulk data bulk-data-id of the bulk data instance to be deleted
	metaInfo	MetaInfo of the request.
	cancellationHandler	Inform if the current conversation is cancelled
Return value:	ara::core::Future< void >	A Future with either a void data result for a successful deletion or a SOVD error code.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for SOVD method "Delete Specific Bulk Data Resource".	

]

8.53.1.1.3.3 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:	<code>class ara::diag::SovdBulkData</code>	
Syntax:	<pre>virtual ara::core::Future< ara::core::Vector< BulkDataMeta-data Descriptor > > GetBulkDataMeta-data (ara::core::String bulkData Category, MetaInfo const &metaInfo, CancellationHandler cancellation Handler) noexcept=0;</pre>	
Parameters (in):	bulkDataCategory	Bulk data category to be read
	metaInfo	MetaInfo of the request.
	cancellationHandler	Inform if the current conversation is cancelled





Return value:	ara::core::Future< ara::core::Vector< Bulk DataMetaDataDescriptor > >	A Future with a Vector of all bulk data instances of the requested bulk data category.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	Called for SOVD method "Read Bulk Data Meta Data".	

]

8.53.1.1.3.4 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		monitor configuration does not match DEXT
	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.53.1.1.3.5 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, BulkDataMetaDataMember >, 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	MetaInfo of the request.
	cancellationHandler	Inform if the current conversation is cancelled
Return value:	ara::core::Future< ara::core::Variant< std::tuple< DataTransferReadSession, BulkDataMetaDataMember >, HttpRedirect > >	A Future with either an instance of a "SOVD file reading session" with the corresponding BulkDataMetaDataMember, an HTTP redirect or a SOVD error code.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1, using DiagSovdErrc .	
Description:	Called for SOVD method "Download Bulk Data".	

]

8.53.1.1.3.6 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, BulkDataMetaDataMember > > RequestBulkDataUpload (ara::core::String bulkDataCategory, BulkDataMetaDataMember bulkDataMetaDataMember, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
Parameters (in):	bulkDataCategory	Bulk data category of which a new bulk data instance shall be uploaded
	bulkDataMetaDataMember	BulkDataMetaDataMember where the available information from the SOVD POST Request headers are prefilled
	metaInfo	MetaInfo of the request.
	cancellationHandler	Inform if the current conversation is cancelled





Return value:	ara::core::Future<std::tuple< DataTransferWriteSession, BulkDataMetaDataDescriptor > >	A Future with either an instance of a "SOVD file writing session" with the corresponding BulkDataMetaDataDescriptor or a SOVD error code.
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for SOVD method "Upload Bulk Data".	

]

8.53.1.1.3.7 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.

]

8.53.1.2 Protected Member Functions

8.53.1.2.1 Special Member Functions

8.53.1.2.1.1 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:	thread-safe	
Description:	Move constructs an instance of SovdBulkData.	
Visibility:	protected	

]

8.53.1.2.1.2 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	

]

8.53.1.2.2 Constructors

8.53.1.2.2.1 SovdBulkData

[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

]

8.53.1.2.3 Member Functions

8.53.1.2.3.1 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

]

8.53.2 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.

]

8.53.2.1 Public Member Variables

8.53.2.1.1 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	creation_date
Type:	ara::core::String
Syntax:	ara::core::String creation_date;
Description:	Specifies the bulk data creation_date.

8.53.2.1.2 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	file_name
Type:	ara::core::String
Syntax:	ara::core::String file_name;
Description:	Specifies the filename.

8.53.2.1.3 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.

]

8.53.2.1.4 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.

]

8.53.2.1.5 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	id
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String id;</code>
Description:	Specifies the bulk data id.

]

8.53.2.1.6 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	last_modified
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String last_modified;</code>
Description:	Specifies the bulk data last_modified.

]

8.53.2.1.7 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	mimetype
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String mimetype;</code>
Description:	Specifies the bulk data mimetype.

]

8.53.2.1.8 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:	<code>struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</code>
Symbol:	name
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String name;</code>
Description:	Specifies the bulk data name.

]

8.53.2.1.9 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.

]

8.53.2.1.10 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.

]

8.53.3 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:	<code>class ara::diag::SovdBulkData</code>
Symbol:	DeleteByCategoryResult
Syntax:	<code>struct DeleteByCategoryResult {...};</code>
Description:	Definition of the result type for the "Delete All Bulk Data Defined by Category" method.

]

8.53.3.1 Public Member Variables

8.53.3.1.1 deleted_items

[SWS_DM_01842] Definition of API variable ara::diag::SovdBulkData::DeleteByCategoryResult::deleted_items

Upstream requirements: [RS_Diag_04266](#)

[

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	<code>struct ara::diag::SovdBulkData::DeleteByCategoryResult</code>
Symbol:	deleted_items
Type:	<code>ara::core::Vector< BulkDataMetaDescriptor ></code>
Syntax:	<code>ara::core::Vector<BulkDataMetaDescriptor> deleted_items;</code>
Description:	List of bulk data ids that were successfully deleted.

]

8.53.3.1.2 failed_items

[SWS_DM_01843] Definition of API variable ara::diag::SovdBulkData::DeleteByCategoryResult::failed_items

Upstream requirements: [RS_Diag_04266](#)

[

Kind:	variable
Header file:	#include "ara/diag/sovd_bulk_data.h"
Scope:	<code>struct ara::diag::SovdBulkData::DeleteByCategoryResult</code>
Symbol:	failed_items
Type:	<code>ara::core::Vector< DeletionError ></code>
Syntax:	<code>ara::core::Vector<DeletionError> failed_items;</code>
Description:	A list of DeletionErrors that contains the bulk data instances with corresponding deletion errors.

]

8.53.4 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:	<code>class ara::diag::SovdBulkData</code>
Symbol:	DeletionError
Syntax:	<code>struct DeletionError {...};</code>
Description:	Definition of DeletionError for a single bulk-data instance.

]

8.53.4.1 Public Member Variables

8.53.4.1.1 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:	<code>struct ara::diag::SovdBulkData::DeletionError</code>
Symbol:	error
Type:	<code>ara::core::ErrorCode</code>
Syntax:	<code>ara::core::ErrorCode error;</code>
Description:	Corresponding error, that specifies why the bulk data could not be deleted.

]

8.53.4.1.2 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:	<code>struct ara::diag::SovdBulkData::DeletionError</code>
Symbol:	item
Type:	<code>BulkDataMetaDataDescriptor</code>
Syntax:	<code>BulkDataMetaDataDescriptor item;</code>
Description:	bulk data that could not be deleted.

]

8.54 Header: ara/diag/sovd_configuration.h

8.54.1 Class: SovdConfiguration

[SWS_DM_01860] Definition of API class ara::diag::SovdConfiguration

Upstream requirements: [RS_Diag_04261](#)

[

Kind:	class
Port Interfaces:	DiagnosticSovdConfigurationInterface
Header file:	#include "ara/diag/sovd_configuration.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	SovdConfiguration
Syntax:	<code>class SovdConfiguration {...};</code>
Description:	Generic SOVD Configuration interface.

]

8.54.1.1 Public Member Functions

8.54.1.1.1 Special Member Functions

8.54.1.1.1.1 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:	<code>virtual ~SovdConfiguration () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdConfiguration.

]

8.54.1.1.2 Constructors

8.54.1.1.2.1 SovdConfiguration

[SWS_DM_01864] Definition of API function `ara::diag::SovdConfiguration::SovdConfiguration`

Upstream requirements: [RS_Diag_04261](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

Kind:	function	
Header file:	#include "ara/diag/sovd_configuration.h"	
Scope:	<code>class ara::diag::SovdConfiguration</code>	
Syntax:	<code>SovdConfiguration (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
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 (kNotConcurrent).
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.
	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.
	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.
Description:	Constructor for the SovdConfiguration Interface.	

8.54.1.1.3 Member Functions

8.54.1.1.3.1 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:	<code>class ara::diag::SovdConfiguration</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	





Return value:	ara::core::Result< void >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		monitor configuration does not match DEXT
	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.54.1.1.3.2 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:	<code>class ara::diag::SovdConfiguration</code>	
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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for SOVD method "Read Configuration".	

]

8.54.1.1.3.3 RequestPutConfiguration

[SWS_DM_01871] Definition of API function ara::diag::SovdConfiguration::RequestPutConfigurationUpstream 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, Sovd ConfigurationMetaInfo > > 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for SOVD method "Write Configuration".	

]

8.54.1.1.3.4 StopOffer

[SWS_DM_01930] Definition of API function ara::diag::SovdConfiguration::Stop OfferUpstream 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.	

]

8.54.1.2 Protected Member Functions

8.54.1.2.1 Special Member Functions

8.54.1.2.1.1 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:	<code>class ara::diag::SovdConfiguration</code>	
Syntax:	<code>SovdConfiguration (SovdConfiguration &&other) noexcept;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move constructs an instance of SovdConfiguration.	
Visibility:	protected	

]

8.54.1.2.1.2 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:	<code>class ara::diag::SovdConfiguration</code>	
Syntax:	<code>SovdConfiguration & operator= (SovdConfiguration &&other) & noexcept;</code>	
Parameters (out):	other	The other object
Return value:	<code>SovdConfiguration &</code>	Reference to self
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assigns an instance of SovdConfiguration.	
Visibility:	protected	

]

8.54.1.2.2 Constructors

8.54.1.2.2.1 SovdConfiguration

[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:	<code>class ara::diag::SovdConfiguration</code>
Syntax:	<code>SovdConfiguration (SovdConfiguration const &)=delete;</code>
Description:	SovdConfiguration shall be a single not copy-able instance.
Visibility:	protected

]

8.54.1.2.3 Member Functions

8.54.1.2.3.1 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:	<code>class ara::diag::SovdConfiguration</code>
Syntax:	<code>SovdConfiguration & operator= (SovdConfiguration const &)=delete;</code>
Description:	SovdConfiguration shall be a single not assignable instance.
Visibility:	protected

]

8.54.2 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.

]

8.54.2.1 Public Member Variables

8.54.2.1.1 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 .

]

8.54.2.1.2 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:	<code>ara::core::Optional< std::uint64_t ></code>
Syntax:	<code>ara::core::Optional<std::uint64_t> size;</code>
Description:	Specifies the size of the configuration data .

]

8.55 Header: `ara/diag/sovd_http_redirect.h`

8.55.1 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 <code>ara::diag</code>
Symbol:	<code>HttpRedirect</code>
Syntax:	<code>struct HttpRedirect {...};</code>
Description:	Definition of HTTP -Redirect. In case a SOVD resource has been moved and/or is available under another URI , the SOVD server shall use this type to indicate the new location of the redirected SOVD resource to the SOVD client.

]

8.55.1.1 Public Member Variables

8.55.1.1.1 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:	<code>struct ara::diag::HttpRedirect</code>
Symbol:	url
Type:	ara::core::String
Syntax:	ara::core::String url;
Description:	Specifies the URL where the request shall be redirected.

]

8.56 Header: ara/diag/sovd_proximity_challenge.h

8.56.1 Non-Member Types

8.56.1.1 Enumeration: ProximityChallengeResultType

[SWS_DM_02382] Definition of API enum ara::diag::ProximityChallengeResultType

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	enumeration	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	ProximityChallengeResultType	
Underlying type:	std::uint8_t	
Syntax:	enum class ProximityChallengeResultType : std::uint8_t {...};	
Values:	kProximityChallengeValidationSuccessful	= 0x00
		proximity_challenge validation successful
	kProximityChallengeValidationFailed	= 0x01
		proximity_challenge validation failed
Description:	This is the type of ProximityChallengeResultType.	

]

8.56.1.2 Type Alias: ProximityChallengeValidationResultType

[SWS_DM_02387] Definition of API type ara::diag::ProximityChallengeValidationResultType

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	type alias
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	namespace ara::diag
Symbol:	ProximityChallengeValidationResultType
Syntax:	using ProximityChallengeValidationResultType = ProximityChallengeResultType;
Description:	This is the type of ProximityChallengeValidationResultType.

]

8.56.2 Class: SovdProximityChallenge

[SWS_DM_01481] Definition of API class ara::diag::SovdProximityChallenge

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	class
Port Interfaces:	DiagnosticSovdProximityChallengeInterface
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.

]

8.56.2.1 Public Member Functions

8.56.2.1.1 Special Member Functions

8.56.2.1.1.1 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:	<code>class ara::diag::SovdProximityChallenge</code>
Syntax:	<code>virtual ~SovdProximityChallenge () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdProximityChallenge.

]

8.56.2.1.2 Constructors

8.56.2.1.2.1 SovdProximityChallenge

[SWS_DM_01480] Definition of API function `ara::diag::SovdProximityChallenge::SovdProximityChallenge`

Upstream requirements: [RS_Diag_04262](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/sovd_proximity_challenge.h"	
Scope:	<code>class ara::diag::SovdProximityChallenge</code>	
Syntax:	<code>explicit SovdProximityChallenge (ara::core::InstanceSpecifier instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a DiagnosticSovdProximityChallengeInterface service instance in the manifest.
	concurrencyType	Specifies if the interface is implemented as thread safe (kConcurrent) or non-thread safe (kNotConcurrent).
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.





	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.
	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.
Description:		Constructs the port for SOVD proximity_challenge interface.

8.56.2.1.3 Member Functions

8.56.2.1.3.1 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	This function returns a proximity_challenge for the client.	

8.56.2.1.3.2 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlready Offered	rollback_semantics
		The service is already offered.
	DiagOfferErrc::k ConfigurationMismatch	rollback_semantics
		monitor configuration does not match DEXT
Description:	This Offer will enable the DM to forward request messages to this handler.	

8.56.2.1.3.3 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.	

8.56.2.1.3.4 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< ProximityChallengeValidationResultType > 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< ProximityChallengeValidationResultType >	Value if the response was valid
Exception Safety:	exception safe	
Thread Safety:	conditional	whether the function has to be implemented thread safe or non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	This function validates the proximity response sent by the client.	

]

8.56.3 Struct: SovdProximityChallengeType

[SWS_DM_01482] Definition of API class ara::diag::SovdProximityChallengeType

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()

]

8.56.3.1 Public Member Variables

8.56.3.1.1 challenge

[SWS_DM_01492] Definition of API variable ara::diag::SovdProximityChallengeType::challenge

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	variable
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	<code>struct ara::diag::SovdProximityChallengeType</code>
Symbol:	challenge
Type:	<code>ara::core::Span< ara::core::Byte ></code>
Syntax:	<code>ara::core::Span<ara::core::Byte> challenge;</code>
Description:	Challenge which has to be solved by the SOVD client to prove proximity.

]

8.56.3.1.2 valid_until

[SWS_DM_01491] Definition of API variable ara::diag::SovdProximityChallengeType::valid_until

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	variable
Header file:	#include "ara/diag/sovd_proximity_challenge.h"
Scope:	<code>struct ara::diag::SovdProximityChallengeType</code>
Symbol:	valid_until
Type:	<code>std::chrono::system_clock::time_point</code>
Syntax:	<code>std::chrono::system_clock::time_point valid_until;</code>
Description:	The time until the challenge is valid.

]

8.57 Header: ara/diag/sovd_service_validation.h

8.57.1 Non-Member Types

8.57.1.1 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:	std::uint8_t	
Syntax:	enum class SovdRequestMethod : std::uint8_t {...};	
Values:	kGet	GET.
	kPut	PUT.
	kPost	POST.
	kDelete	DELETE.
Description:	HTTP method used in the SOVD request.	

]

8.57.2 Class: SovdServiceValidation

[SWS_DM_01822] Definition of API class ara::diag::SovdServiceValidation

Upstream requirements: [RS_Diag_04256](#)

[

Kind:	class	
Port Interfaces:	DiagnosticSovdServiceValidationInterface	
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.	

]

8.57.2.1 Public Member Functions

8.57.2.1.1 Special Member Functions

8.57.2.1.1.1 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:	<code>class ara::diag::SovdServiceValidation</code>
Syntax:	<code>virtual ~SovdServiceValidation () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of SovdServiceValidation.

]

8.57.2.1.2 Constructors

8.57.2.1.2.1 SovdServiceValidation

[SWS_DM_01823] Definition of API function `ara::diag::SovdServiceValidation::SovdServiceValidation`

Upstream requirements: [RS_Diag_04256](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/sovd_service_validation.h"	
Scope:	<code>class ara::diag::SovdServiceValidation</code>	
Syntax:	<code>explicit SovdServiceValidation (ara::core::InstanceSpecifier instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
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 (kNotConcurrent).
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.





	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.
	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.
Description:		Constructs the port for SovdServiceValidation interface.

8.57.2.1.3 Member Functions

8.57.2.1.3.1 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:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		The service is already offered.
	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		monitor configuration does not match DEXT
Description:	Enable the DM to forward request messages to this handler.	

8.57.2.1.3.2 StopOffer

[SWS_DM_01826] Definition of API function ara::diag::SovdServiceValidation::StopOfferUpstream 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.

]

8.57.2.1.3.3 Validate

[SWS_DM_01827] Definition of API function ara::diag::SovdServiceValidation::ValidateUpstream 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 request Method, 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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	DiagSovdErrc::kSovdServerMisconfigured	rollback_semantics
		The SOVD server is not configured correctly, e.g. required configuration files or other data is missing.
	DiagSovdErrc::kPreconditionNotFulfilled	rollback_semantics
		The preconditions to execute the method are not fulfilled.
	DiagSovdErrc::kTemporarilyNotAvailable	rollback_semantics

▽



		The request currently can't be served, e.g. 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).
	DiagSovdErrc::kInsufficientAccessRights	rollback_semantics
		The SOVD client does not have the right to access the resource.
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.	
Notes:	Or any other DiagSovdErrc for an SOVD-specific error message.	

]

8.58 Header: ara/diag/sovд_sw_update.h

8.58.1 Non-Member Types

8.58.1.1 Enumeration: SovdUpdatePhase

[SWS_DM_01875] Definition of API enum ara::diag::SovdUpdatePhase

Upstream requirements: [RS_Diag_04265](#)

[

Kind:	enumeration	
Header file:	#include "ara/diag/sovд_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	The Update is in preparation.
	kExecute	The Update is in execution.
Description:	Phase type for a SOVD update.	

]

8.58.1.2 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 .	

8.58.2 Class: SovdSwUpdate

[SWS_DM_01874] Definition of API class ara::diag::SovdSwUpdate

Upstream requirements: [RS_Diag_04265](#)

Kind:	class
Port Interfaces:	DiagnosticSovdUpdateInterface
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.

8.58.2.1 Public Member Functions

8.58.2.1.1 Special Member Functions

8.58.2.1.1.1 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:	<code>class ara::diag::SovdSwUpdate</code>
Syntax:	<code>virtual ~SovdSwUpdate () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of class SovdSwUpdate.

]

8.58.2.1.2 Constructors

8.58.2.1.2.1 SovdSwUpdate

[SWS_DM_01910] Definition of API function `ara::diag::SovdSwUpdate::SovdSwUpdate`

Upstream requirements: [RS_Diag_04265](#), [RS_AP_00170](#), [RS_AP_00171](#), [RS_AP_00172](#), [RS_AP_00173](#)

[

Kind:	function	
Header file:	#include "ara/diag/sovd_sw_update.h"	
Scope:	<code>class ara::diag::SovdSwUpdate</code>	
Syntax:	<code>SovdSwUpdate (ara::core::InstanceSpecifier const &instanceSpecifier, ConcurrencyType concurrencyType) noexcept;</code>	
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 (kNotConcurrent).
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.





	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.
	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.
Description:	Constructor for SovdSwUpdate.	

]

8.58.2.1.3 Member Functions

8.58.2.1.3.1 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Delete Update Package from an SOVD Server".	

]

8.58.2.1.3.2 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Execute Installation of an Update".	

]

8.58.2.1.3.3 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 cancellationHandler) 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled





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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	This function does not specify any standardized errors.	
Description:	Called for SOVD method "Retrieve List of All Updates".	

]

8.58.2.1.3.4 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Get Details of Update".	

]

8.58.2.1.3.5 GetUpdatePackageStatus

[SWS_DM_01921] Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackageStatusUpstream 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Get Status of an Update".	

]

8.58.2.1.3.6 Offer

[SWS_DM_01924] Definition of API function ara::diag::SovdSwUpdate::OfferUpstream 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 >	Returns nothing or an error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	DiagOfferErrc::kConfigurationMismatch	rollback_semantics
		monitor configuration does not match DEXT
	DiagOfferErrc::kAlreadyOffered	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.58.2.1.3.7 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Prepare Installation of an Update".	

]

8.58.2.1.3.8 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	MetaInfo of the request.
	cancellationHandler	Informs if the current conversation is cancelled
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 non-thread safe is determined by the ConcurrencyType given in the constructor





Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.
Description:	Called for SOVD method "Automated Installation of an Update".

]

8.58.2.1.3.9 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:	<code>class ara::diag::SovdSwUpdate</code>	
Syntax:	<pre>virtual ara::core::Future< std::tuple< DataTransferWriteSession, ara::core::String > > RequestUpdatePackageRegistration (MetaInfo const &metaInfo, ara::core::Optional< uint64_t > expectedSize, Cancellation Handler cancellationHandler) noexcept=0;</pre>	
Parameters (in):	metaInfo	MetaInfo of the request.
	expectedSize	Specifies the expected size if available in the request.
	cancellationHandler	Informs if the current conversation is cancelled
Return value:	<pre>ara::core::Future< std::tuple< DataTransfer WriteSession, ara::core::String > ></pre>	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 non-thread safe is determined by the <code>ConcurrencyType</code> given in the constructor
Errors:	Please use applicable error code from ASAM SOVD 1.1 or ISO 17978-3.	
Description:	Called for SOVD method "Register an Update at the SOVD Server".	

]

8.58.2.1.3.10 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:	<code>class ara::diag::SovdSwUpdate</code>
Syntax:	<code>void StopOffer () const noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.58.2.2 Protected Member Functions

8.58.2.2.1 Special Member Functions

8.58.2.2.1.1 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:	<code>class ara::diag::SovdSwUpdate</code>		
Syntax:	<code>SovdSwUpdate (SovdSwUpdate &&other) noexcept;</code>		
Parameters (out):	<table><tr><td>other</td><td>The other object</td></tr></table>	other	The other object
other	The other object		
Exception Safety:	exception safe		
Thread Safety:	unsafe		
Description:	Move constructs an instance of SovdSwUpdate.		
Visibility:	protected		

]

8.58.2.2.1.2 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	

]

8.58.2.2.2 Constructors

8.58.2.2.2.1 SovdSwUpdate

[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	

]

8.58.2.2.3 Member Functions

8.58.2.2.3.1 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:	<code>class ara::diag::SovdSwUpdate</code>
Syntax:	<code>SovdSwUpdate & operator= (SovdSwUpdate const &)=delete;</code>
Description:	SovdSwUpdate shall be a single not assignable instance.
Visibility:	protected

]

8.58.3 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:	<code>class ara::diag::SovdSwUpdate</code>
Symbol:	SubProgress
Syntax:	<code>struct SubProgress {...};</code>
Description:	Definition of Sovd Progress type.

]

8.58.3.1 Public Member Variables

8.58.3.1.1 entity

[SWS_DM_01898] Definition of API variable `ara::diag::SovdSwUpdate::SubProgress::entity`

Upstream requirements: [RS_Diag_04265](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/sovd_sw_update.h"</code>
Scope:	<code>struct ara::diag::SovdSwUpdate::SubProgress</code>
Symbol:	entity
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String entity;</code>
Description:	Specifies the entity that is currently being updated.

]

8.58.3.1.2 error

[SWS_DM_01901] Definition of API variable `ara::diag::SovdSwUpdate::SubProgress::error`

Upstream requirements: [RS_Diag_04265](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/sovd_sw_update.h"</code>
Scope:	<code>struct ara::diag::SovdSwUpdate::SubProgress</code>
Symbol:	error
Type:	<code>ara::core::ErrorCode</code>
Syntax:	<code>ara::core::ErrorCode error;</code>
Description:	Specifies the error if status is failed.

]

8.58.3.1.3 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.

]

8.58.3.1.4 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.

]

8.58.4 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 .

]

8.58.4.1 Public Member Variables

8.58.4.1.1 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.

]

8.58.4.1.2 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:	<code>ara::core::Optional< bool ></code>
Syntax:	<code>ara::core::Optional<bool> automated;</code>
Description:	Specifies the update package attribute "automated".

]

8.58.4.1.3 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:	<code>ara::core::Optional< std::uint64_t ></code>
Syntax:	<code>ara::core::Optional<std::uint64_t> duration;</code>
Description:	Specifies the update package duration in seconds.

]

8.58.4.1.4 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.

]

8.58.4.1.5 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.

]

8.58.4.1.6 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageDetails</code>
Symbol:	name
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String name;</code>
Description:	Specifies the update package update_name.

]

8.58.4.1.7 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageDetails</code>
Symbol:	notes
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String notes;</code>
Description:	Specifies the update package notes.

]

8.58.4.1.8 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.

]

8.58.4.1.9 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.

]

8.58.4.1.10 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.

]

8.58.4.1.11 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.

]

8.58.4.1.12 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 .

]

8.58.4.1.13 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.

]

8.58.4.1.14 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.

]

8.58.4.1.15 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.

]

8.58.4.1.16 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.

]

8.58.5 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.

]

8.58.5.1 Public Member Variables

8.58.5.1.1 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageStatus</code>
Symbol:	error
Type:	<code>ara::core::Optional< ara::diag::DiagSovdErrc ></code>
Syntax:	<code>ara::core::Optional<ara::diag::DiagSovdErrc> error;</code>
Description:	Specifies the error if status is failed.

]

8.58.5.1.2 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageStatus</code>
Symbol:	phase
Type:	<code>SovdUpdatePhase</code>
Syntax:	<code>SovdUpdatePhase phase;</code>
Description:	Specifies the Update Status phase.

]

8.58.5.1.3 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.

]

8.58.5.1.4 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.

]

8.58.5.1.5 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageStatus</code>
Symbol:	step
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String step;</code>
Description:	Specifies the current step of the Update.

]

8.58.5.1.6 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:	<code>struct ara::diag::SovdSwUpdate::UpdatePackageStatus</code>
Symbol:	step_translation_id
Type:	<code>ara::core::String</code>
Syntax:	<code>ara::core::String step_translation_id;</code>
Description:	Specifies the step_translation_id.

]

8.58.5.1.7 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:	<code>ara::core::Vector< SubProgress ></code>
Syntax:	<code>ara::core::Vector<SubProgress> subprogress;</code>
Description:	Specifies the progress of the Update phase in percent.

]

8.59 Header: `ara/diag/diagnostic_sovd_content_data.h`

8.59.1 Struct: `DiagnosticSovdContentData`

[SWS_DM_02385] Definition of API class `ara::diag::DiagnosticSovdContentData`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	struct
Header file:	#include "ara/diag/diagnostic_sovd_content_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace <code>ara::diag</code>
Symbol:	<code>DiagnosticSovdContentData</code>
Syntax:	<code>struct DiagnosticSovdContentData {...};</code>
Description:	The <code>DiagnosticSovdContentData</code> declaration.

]

8.59.1.1 Public Member Variables

8.59.1.1.1 elements

[SWS_DM_02386] Definition of API variable `ara::diag::DiagnosticSovdContentData::elements`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	variable
Header file:	<code>#include "ara/diag/diagnostic_sovd_content_data.h"</code>
Scope:	<code>struct ara::diag::DiagnosticSovdContentData</code>
Symbol:	<code>elements</code>
Type:	<code>ara::core::Vector< DiagnosticSovdContentElementData ></code>
Syntax:	<code>ara::core::Vector<DiagnosticSovdContentElementData> elements;</code>
Description:	The diagnostic sovd content elements.

]

8.60 Header: `ara/diag/diagnostic_sovd_content_element_data.h`

8.60.1 Non-Member Types

8.60.1.1 Enumeration: `DiagnosticSovdContentElementType`

[SWS_DM_02213] Definition of API enum `ara::diag::DiagnosticSovdContentElementType`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	enumeration	
Header file:	<code>#include "ara/diag/diagnostic_sovd_content_element_data.h"</code>	
Forwarding header file:	<code>#include "ara/diag/diag_fwd.h"</code>	
Scope:	namespace <code>ara::diag</code>	
Symbol:	<code>DiagnosticSovdContentElementType</code>	
Underlying type:	<code>std::uint8_t</code>	
Syntax:	<code>enum class DiagnosticSovdContentElementType : std::uint8_t {...};</code>	
Values:	<code>kContentData</code>	type <code>ContentData</code>
	<code>kPrimitiveContentElementData</code>	type <code>PrimitiveContentData</code>
	<code>kArrayContentElementData</code>	type Array of <code>ContentElementData</code>
	<code>kRecordContentElementData</code>	type Record of <code>ContentElementData</code>
Description:	The type of the SOVD content data.	

]

8.60.1.2 Enumeration: DiagnosticSovdPrimitiveContentType**[SWS_DM_02214] Definition of API enum ara::diag::DiagnosticSovdPrimitiveContentType***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	enumeration	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdPrimitiveContentType	
Underlying type:	std::uint8_t	
Syntax:	enum class DiagnosticSovdPrimitiveContentType : std::uint8_t { ... };	
Values:	kSignedInt	type signed integer
	kUnsignedInt	type unsigned integer
	kDouble	type double
	kBool	type boolean
	kString	type string
Description:	The primitive content element value types.	

]

8.60.2 Class: DiagnosticSovdArrayContentElementDataConstIterator**[SWS_DM_02221] Definition of API class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	class	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdArrayContentElementDataConstIterator	
Syntax:	class DiagnosticSovdArrayContentElementDataConstIterator { ... };	
Description:	Const iterator for a DiagnosticSovdArrayContentElementData.	

]

8.60.2.1 Public Member Functions

8.60.2.1.1 Member Functions

8.60.2.1.1.1 operator!=

[SWS_DM_02225] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstIterator::operator!=`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator</code>	
Syntax:	<code>auto operator!= (DiagnosticSovdArrayContentElementDataConstIterator const &other) noexcept -> bool;</code>	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Not equal operator.	

]

8.60.2.1.1.2 operator*

[SWS_DM_02222] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstIterator::operator*`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator</code>	
Syntax:	<code>auto operator* () noexcept -> DiagnosticSovdContentElementData const &;</code>	
Return value:	DiagnosticSovdContentElementData const	Return a reference to the DiagnosticSovdContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.2.1.1.3 operator++

[SWS_DM_02226] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstIterator::operator++

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator	
Syntax:	auto operator++ () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator &;	
Return value:	DiagnosticSovdArrayContentElementDataConstIterator	Reference to the incremented iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Increment operator.	

]

8.60.2.1.1.4 operator->

[SWS_DM_02223] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstIterator::operator->

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator	
Syntax:	auto operator-> () noexcept -> DiagnosticSovdContentElementData const *;	
Return value:	DiagnosticSovdContentElementData const *	Return a pointer to the DiagnosticSovdContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.2.1.1.5 operator==

[SWS_DM_02224] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstIterator::operator==

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstIterator	
Syntax:	auto operator== (DiagnosticSovdArrayContentElementDataConstIterator const &other) noexcept -> bool;	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Equal operator.	

]

8.60.3 Class: DiagnosticSovdArrayContentElementDataConstView

[SWS_DM_02248] Definition of API class ara::diag::DiagnosticSovdArrayContentElementDataConstView

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdArrayContentElementDataConstView	
Syntax:	class DiagnosticSovdArrayContentElementDataConstView {...};	
Description:	Const view on a DiagnosticSovdContentElementData with Array type.	

]

8.60.3.1 Public Member Functions

8.60.3.1.1 Member Functions

8.60.3.1.1.1 Begin

[SWS_DM_02254] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::Begin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataConstView</code>	
Syntax:	<code>auto Begin () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdArray ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.3.1.1.2 CBegin

[SWS_DM_02256] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::CBegin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataConstView</code>	
Syntax:	<code>auto CBegin () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdArray ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.3.1.1.3 CEnd

[SWS_DM_02257] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstView::CEnd*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto CEnd () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator ;	
Return value:	DiagnosticSovdArrayContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.3.1.1.4 Capacity

[SWS_DM_02251] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstView::Capacity*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto Capacity () const noexcept -> std::size_t;	
Return value:	std::size_t	The capacity of the ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.3.1.1.5 Empty

[SWS_DM_02252] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::Empty`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto Empty () const noexcept -> bool;	
Return value:	bool	If the array is empty
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the Array is empty. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.3.1.1.6 End

[SWS_DM_02255] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::End`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto End () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator ;	
Return value:	DiagnosticSovdArray ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.3.1.1.7 HasError

[SWS_DM_02249] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::HasError`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto HasError () const noexcept -> ara::core::Result< void >;	
Return value:	ara::core::Result< void >	void result if it doesn't contain an error, otherwise the ara::diag::DiagSovdErrc
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the content contains an error.	

]

8.60.3.1.1.8 Size

[SWS_DM_02250] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataConstView::Size`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	auto Size () const noexcept -> std::size_t;	
Return value:	std::size_t	The size of the ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the size of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.3.1.1.9 operator[]

[SWS_DM_02253] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstView::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataConstView	
Syntax:	<pre>auto operator[] (std::size_t index) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > >;</pre>	
DIRECTION NOT DEFINED	index	--
Return value:	<pre>ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > ></pre>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified index of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.4 Class: DiagnosticSovdArrayContentElementDataIterator

[SWS_DM_02215] Definition of API class ara::diag::DiagnosticSovdArrayContentElementDataIterator

Upstream requirements: [RS_Diag_04260](#)

Kind:	class
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DiagnosticSovdArrayContentElementDataIterator
Syntax:	<pre>class DiagnosticSovdArrayContentElementDataIterator {...};</pre>
Description:	Iterator for a DiagnosticSovdArrayContentElementData.

8.60.4.1 Public Member Functions

8.60.4.1.1 Member Functions

8.60.4.1.1.1 operator!=

[SWS_DM_02219] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataIterator::operator!=`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataIterator</code>	
Syntax:	<code>auto operator!= (DiagnosticSovdArrayContentElementDataIterator const &other) noexcept -> bool;</code>	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Not equal operator.	

]

8.60.4.1.1.2 operator*

[SWS_DM_02216] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataIterator::operator*`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataIterator</code>	
Syntax:	<code>auto operator* () noexcept -> DiagnosticSovdContentElementData &;</code>	
Return value:	DiagnosticSovdContentElementData	Return a reference to the DiagnosticSovdContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.4.1.1.3 operator++

[SWS_DM_02220] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataIterator::operator++

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataIterator	
Syntax:	auto operator++ () noexcept -> DiagnosticSovdArrayContentElementDataIterator &;	
Return value:	DiagnosticSovdArrayContentElementDataIterator	Reference to the incremented iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Increment operator.	

]

8.60.4.1.1.4 operator->

[SWS_DM_02217] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataIterator::operator->

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataIterator	
Syntax:	auto operator-> () noexcept -> DiagnosticSovdContentElementData *;	
Return value:	DiagnosticSovdContentElementData *	Return a pointer to the DiagnosticSovdContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.4.1.1.5 operator==

[SWS_DM_02218] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataIterator::operator==

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataIterator	
Syntax:	auto operator== (DiagnosticSovdArrayContentElementDataIterator const &other) noexcept -> bool;	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Equal operator.	

]

8.60.5 Class: DiagnosticSovdArrayContentElementDataView

[SWS_DM_02227] Definition of API class ara::diag::DiagnosticSovdArrayContentElementDataView

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdArrayContentElementDataView	
Syntax:	class DiagnosticSovdArrayContentElementDataView {...};	
Description:	View on a DiagnosticSovdContentElementData with Array type.	

]

8.60.5.1 Public Member Functions

8.60.5.1.1 Member Functions

8.60.5.1.1.1 Begin

[SWS_DM_02236] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Begin`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto Begin () noexcept -> DiagnosticSovdArrayContentElementData Iterator;</code>	
Return value:	DiagnosticSovdArray ContentElementData Iterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.2 CBegin

[SWS_DM_02238] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::CBegin`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto CBegin () noexcept -> DiagnosticSovdArrayContentElementDataConst Iterator;</code>	
Return value:	DiagnosticSovdArray ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.3 CEnd

[SWS_DM_02239] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::CEnd`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto CEnd () noexcept -> DiagnosticSovdArrayContentElementDataConstIterator ;	
Return value:	DiagnosticSovdArrayContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.4 Capacity

[SWS_DM_02231] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Capacity`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto Capacity () const noexcept -> std::size_t;	
Return value:	std::size_t	The capacity of the ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.5 Clear

[SWS_DM_02233] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Clear`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>
Syntax:	<code>auto Clear () noexcept -> void;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Clears the Array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData

]

8.60.5.1.1.6 Empty

[SWS_DM_02232] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Empty`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>
Syntax:	<code>auto Empty () const noexcept -> bool;</code>
Return value:	bool If the array is empty
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Returns if the Array is empty. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData

]

8.60.5.1.1.7 End

[SWS_DM_02237] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::End

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto End () noexcept -> DiagnosticSovdArrayContentElementDataIterator ;	
Return value:	DiagnosticSovdArray ContentElementData Iterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.8 Erase

[SWS_DM_02247] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::Erase

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto Erase (DiagnosticSovdArrayContentElementDataIterator iterator) noexcept -> ara::core::Result< void >;	
Parameters (in):	iterator	The iterator pointing to the element to erase
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData or the element couldn't be erased
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Erases a content element from DiagnosticSovdContentElementData. The current element shall be of ArrayContentElementData	

]

8.60.5.1.1.9 HasError

[SWS_DM_02228] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::HasError`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto HasError () const noexcept -> ara::core::Result< void >;</code>	
Return value:	<code>ara::core::Result< void ></code>	void result if it doesn't contain an error, otherwise the <code>ara::diag::DiagSovdErrc</code>
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the content contains an error.	

]

8.60.5.1.1.10 PushBack

[SWS_DM_02246] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::PushBack`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto PushBack (DiagnosticSovdContentElementData content_element) noexcept -> void;</code>	
Parameters (in):	<code>content_element</code>	The content element to push back
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a <code>DiagnosticSovdContentElementData</code> . The current element shall be of <code>ArrayContentElementData</code> and the element to add of the specified in the <code>ArrayContentElementData</code> constructor	

]

8.60.5.1.1.11 PushBack

[SWS_DM_02245] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::PushBackUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto PushBack (ara::core::StringView content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with string as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.5.1.1.12 PushBack

[SWS_DM_02244] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::PushBackUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto PushBack (bool content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with boolean as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.5.1.1.13 PushBack

[SWS_DM_02243] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::PushBack`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto PushBack (double content_element) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	content_element	The content element to push back
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with double as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.5.1.1.14 PushBack

[SWS_DM_02241] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::PushBack`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto PushBack (std::int64_t content_element) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	content_element	The content element to push back
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with signed int as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.5.1.1.15 PushBack

[SWS_DM_02242] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::PushBack`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto PushBack (std::uint64_t content_element) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with unsigned int as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.5.1.1.16 Reserve

[SWS_DM_02240] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Reserve`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto Reserve (std::size_t capacity) noexcept -> void;</code>	
Parameters (in):	capacity	The capacity
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reserves the capacity of the DiagnosticSovdArrayContentElementData.	

]

8.60.5.1.1.17 SetError

[SWS_DM_02229] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::SetError`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	void SetError (ara::core::ErrorCode error_code) noexcept;	
Parameters (in):	error_code	The error to set
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Sets the array content to be erroneous.	

]

8.60.5.1.1.18 Size

[SWS_DM_02230] Definition of API function `ara::diag::DiagnosticSovdArrayContentElementDataView::Size`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdArrayContentElementDataView	
Syntax:	auto Size () const noexcept -> std::size_t;	
Return value:	std::size_t	The size of the ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the size of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.5.1.1.19 operator[]

[SWS_DM_02234] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto operator[] (std::size_t index) noexcept -> ara::core::Result<std::reference_wrapper< DiagnosticSovdContentElementData > >;</code>	
DIRECTION NOT DEFINED	index	--
Return value:	<code>ara::core::Result<std::reference_wrapper< DiagnosticSovdContentElementData > ></code>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified index of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.5.1.1.20 operator[]

[SWS_DM_02235] Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataView::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdArrayContentElementDataView</code>	
Syntax:	<code>auto operator[] (std::size_t index) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData const > >;</code>	
DIRECTION NOT DEFINED	index	--
Return value:	<code>ara::core::Result<std::reference_wrapper< DiagnosticSovdContentElementData const > ></code>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified index of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.6 Class: DiagnosticSovdContentElementData

[SWS_DM_02307] Definition of API class ara::diag::DiagnosticSovdContentElementData

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DiagnosticSovdContentElementData
Syntax:	class DiagnosticSovdContentElementData {...};
Description:	Diagnostic SOVD Abstract Content Data.

]

8.60.6.1 Public Member Functions

8.60.6.1.1 Special Member Functions

8.60.6.1.1.1 Destructor

[SWS_DM_02316] Definition of API function ara::diag::DiagnosticSovdContentElementData::~~DiagnosticSovdContentElementData

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Scope:	class ara::diag::DiagnosticSovdContentElementData
Syntax:	~DiagnosticSovdContentElementData () noexcept;
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Declare the Dtor and move/copy constructors/assignments .

]

8.60.6.1.2 Member Functions

8.60.6.1.2.1 Add

[SWS_DM_02368] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, std::int64_t content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with signed integer as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

8.60.6.1.2.2 Add

[SWS_DM_02369] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, std::uint64_t content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	





Description:	Adds an element with the specified key with unsigned integer as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData

]

8.60.6.1.2.3 Add

[SWS_DM_02372] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Add`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Add (ara::core::StringView key, ara::core::StringView content_element) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with string as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.6.1.2.4 Add

[SWS_DM_02371] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Add`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Add (ara::core::StringView key, bool content_element) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	key	The key
	content_element	The content element to add





Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with boolean as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.6.1.2.5 Add

[SWS_DM_02370] Definition of API function ara::diag::DiagnosticSovdContentElementData::Add

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto Add (ara::core::StringView key, double content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with double as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.6.1.2.6 Add

[SWS_DM_02373] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, DiagnosticSovdContentElementData content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

8.60.6.1.2.7 Begin

[SWS_DM_02341] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Begin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Begin () noexcept -> ara::core::Result< DiagnosticSovdArray ContentElementDataIterator >;</pre>	
Return value:	<code>ara::core::Result< DiagnosticSovdArray ContentElementData Iterator ></code>	The iterator or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.6.1.2.8 CBegin

[SWS_DM_02343] Definition of API function `ara::diag::DiagnosticSovdContentElementData::CBegin`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto CBegin () const noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataConstIterator >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdArrayContentElementDataConstIterator ></code>	The iterator or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.9 CEnd

[SWS_DM_02344] Definition of API function `ara::diag::DiagnosticSovdContentElementData::CEnd`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto CEnd () const noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataConstIterator >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdArrayContentElementDataConstIterator ></code>	The iterator or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	<code>DiagErrc::kInvalidArgument</code>	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.10 Capacity

[SWS_DM_02336] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Capacity`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Capacity () const noexcept -> ara::core::Result< std::size_t >;</code>	
Return value:	<code>ara::core::Result< std::size_t ></code>	The capacity of the ArrayContentElementData or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.11 Clear

[SWS_DM_02338] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Clear`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Clear () noexcept -> ara::core::Result< void >;</code>	
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Clears the Array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.12 CreateArrayContentElement

[SWS_DM_02313] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateArrayContentElementUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreateArrayContentElement (DiagnosticSovdContentElementDataType elements_type) -> DiagnosticSovdContentElementData ;	
Parameters (in):	elements_type	The type of the elements contained in the array. It shall be of type ContentData
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create an array content element.	

]

8.60.6.1.2.13 CreateContentElement

[SWS_DM_02315] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateContentElementUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreateContentElement () -> DiagnosticSovdContentElementData ;	
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create a content.	

]

8.60.6.1.2.14 CreatePrimitiveContentBool

[SWS_DM_02311] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentBool

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreatePrimitiveContentBool (bool value=false) -> DiagnosticSovdContentElementData ;	
Parameters (in):	value	The value to set
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create primitive of type bool.	

]

8.60.6.1.2.15 CreatePrimitiveContentDouble

[SWS_DM_02310] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentDouble

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreatePrimitiveContentDouble (double value=0) -> DiagnosticSovdContentElementData ;	
Parameters (in):	value	The value to set
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create primitive of type double.	

]

8.60.6.1.2.16 CreatePrimitiveContentSignedInt**[SWS_DM_02308] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentSignedInt***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreatePrimitiveContentSignedInt (std::int64_t value=0) -> DiagnosticSovdContentElementData ;	
Parameters (in):	value	The value to set
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create primitive of type signed integer.	

]

8.60.6.1.2.17 CreatePrimitiveContentString**[SWS_DM_02312] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentString***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreatePrimitiveContentString (ara::core::StringView value=ara::core::StringView{}) -> DiagnosticSovdContentElementData ;	
Parameters (in):	value	The value to set
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create primitive of type string.	

]

8.60.6.1.2.18 CreatePrimitiveContentUnsignedInt**[SWS_DM_02309] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentUnsignedInt**

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreatePrimitiveContentUnsignedInt (std::uint64_t value=0) -> DiagnosticSovdContentElementData ;	
Parameters (in):	value	The value to set
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create primitive of type unsigned integer.	

]

8.60.6.1.2.19 CreateRecordContentElement**[SWS_DM_02314] Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateRecordContentElement**

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	static auto CreateRecordContentElement () -> DiagnosticSovdContentElementData ;	
Return value:	DiagnosticSovdContentElementData	The created DiagnosticSovdContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Create a record content element.	

]

8.60.6.1.2.20 Empty

[SWS_DM_02337] Definition of API function ara::diag::DiagnosticSovdContentElementData::EmptyUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto Empty () const noexcept -> ara::core::Result< bool >;	
Return value:	ara::core::Result< bool >	If the array is empty or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns if the Array is empty. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.21 End

[SWS_DM_02342] Definition of API function ara::diag::DiagnosticSovdContentElementData::EndUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto End () noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataIterator >;	
Return value:	ara::core::Result< DiagnosticSovdArrayContentElementDataIterator >	The iterator or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.22 Erase

[SWS_DM_02352] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Erase`*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Erase (DiagnosticSovdArrayContentElementDataIterator iterator) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	iterator	The iterator pointing to the element to erase
Return value:	<code>ara::core::Result< void ></code>	void result or an error if <code>DiagnosticSovdContentElementData</code> is not of type <code>ArrayContentElementData</code> or the element couldn't be erased
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Erases a content element from <code>DiagnosticSovdContentElementData</code> . The current element shall be of <code>ArrayContentElementData</code>	

]

8.60.6.1.2.23 Erase

[SWS_DM_02374] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Erase`*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto Erase (DiagnosticSovdRecordContentElementDataIterator iterator) noexcept -> ara::core::Result< void >;</code>	
Parameters (in):	iterator	The iterator pointing to the element to erase
Return value:	<code>ara::core::Result< void ></code>	void result or an error if <code>DiagnosticSovdContentElementData</code> is not of type <code>RecordContentElementData</code> or <code>ContentData</code> OR the element couldn't be erased
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Erases a content element from <code>DiagnosticSovdRecordContentElementData</code> . The current element shall be of <code>RecordContentElementData</code> or <code>ContentData</code>	

]

8.60.6.1.2.24 Find

[SWS_DM_02366] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Find`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Find (ara::core::StringView key) const noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator >;</pre>	
Parameters (in):	key	The key
Return value:	<code>ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator ></code>	A DiagnosticSovdRecordContentElementDataIterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Find an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.6.1.2.25 Find

[SWS_DM_02365] Definition of API function `ara::diag::DiagnosticSovdContentElementData::Find`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto Find (ara::core::StringView key) noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataIterator >;</pre>	
Parameters (in):	key	The key
Return value:	<code>ara::core::Result< DiagnosticSovdRecordContentElementDataIterator ></code>	A DiagnosticSovdRecordContentElementDataIterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Find an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.6.1.2.26 GetArrayView

[SWS_DM_02333] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetArrayView`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto GetArrayView () const noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataConstView >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdArrayContentElementDataConstView ></code>	A view onto this DiagnosticSovdContentElementData as an array or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get a view onto this DiagnosticSovdContentElementData as an array.	

]

8.60.6.1.2.27 GetArrayView

[SWS_DM_02331] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetArrayView`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto GetArrayView () noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataView >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdArrayContentElementDataView ></code>	A view onto this DiagnosticSovdContentElementData as an array or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get a view onto this DiagnosticSovdContentElementData as an array.	

]

8.60.6.1.2.28 GetBool

[SWS_DM_02329] Definition of API function ara::diag::DiagnosticSovdContentElementData::GetBool*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto GetBool () const noexcept -> ara::core::Result< bool >;	
Return value:	ara::core::Result< bool >	The value or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData with bool value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get bool.	

]

8.60.6.1.2.29 GetDouble

[SWS_DM_02328] Definition of API function ara::diag::DiagnosticSovdContentElementData::GetDouble*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto GetDouble () const noexcept -> ara::core::Result< double >;	
Return value:	ara::core::Result< double >	The value or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData with double value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get double.	

]

8.60.6.1.2.30 GetPrimitiveContentType**[SWS_DM_02320] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetPrimitiveContentType`***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto GetPrimitiveContentType () const noexcept -> ara::core::Result< DiagnosticSovdPrimitive ContentValueType >;</pre>	
Return value:	<code>ara::core::Result< DiagnosticSovdPrimitive ContentValueType ></code>	Returns the underlying value type of the primitive content or an error if this content is not of type <code>PrimitiveContentElementData</code>
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get the underlying value type of the primitive content. This <code>DiagnosticSovdAbstractContentData</code> shall be of type <code>PrimitiveContentElementData</code>	

]

8.60.6.1.2.31 GetRecordView**[SWS_DM_02332] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetRecordView`***Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto GetRecordView () noexcept -> ara::core::Result< DiagnosticSovd ArrayContentElementDataView >;</pre>	
Return value:	<code>ara::core::Result< DiagnosticSovdArray ContentElementData View ></code>	A view onto this <code>DiagnosticSovdContentElementData</code> as a record or an error if <code>DiagnosticSovdContentElementData</code> is not of type <code>ArrayContentElementData</code>
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get a view onto this <code>DiagnosticSovdContentElementData</code> as a record.	

]

8.60.6.1.2.32 GetRecordView

[SWS_DM_02334] Definition of API function ara::diag::DiagnosticSovdContentElementData::GetRecordViewUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto GetRecordView () const noexcept -> ara::core::Result< DiagnosticSovdArrayContentElementDataConstView >;	
Return value:	ara::core::Result< DiagnosticSovdArrayContentElementDataConstView >	A view onto this DiagnosticSovdContentElementData as a record or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get a view onto this DiagnosticSovdContentElementData as a record.	

]

8.60.6.1.2.33 GetSignedInt

[SWS_DM_02326] Definition of API function ara::diag::DiagnosticSovdContentElementData::GetSignedIntUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto GetSignedInt () const noexcept -> ara::core::Result< std::int64_t >;	
Return value:	ara::core::Result< std::int64_t >	The value or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData with signed value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get signed integer.	

]

8.60.6.1.2.34 GetString

[SWS_DM_02330] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetString`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto GetString () -> ara::core::Result< ara::core::StringView >;</code>	
Return value:	<code>ara::core::Result< ara::core::StringView ></code>	The value or error if <code>DiagnosticSovdContentElementData</code> is not of type <code>PrimitiveContentElementData</code> with string value
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Get string. The string view is valid as long a this object is valid and the value is not replaced	

]

8.60.6.1.2.35 GetType

[SWS_DM_02319] Definition of API function `ara::diag::DiagnosticSovdContentElementData::GetType`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto GetType () const noexcept -> DiagnosticSovdContentElementData;</code>	
Return value:	<code>DiagnosticSovdContentElementData</code>	The SOVD content data type
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Get the underlying type of the content.	

]

8.60.6.1.2.36 GetUnsignedInt

[SWS_DM_02327] Definition of API function ara::diag::DiagnosticSovdContentElementData::GetUnsignedIntUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto GetUnsignedInt () const noexcept -> ara::core::Result< std::uint64_t >;	
Return value:	ara::core::Result< std::uint64_t >	The value or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData with unsigned value
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalid Argument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Get unsigned integer.	

]

8.60.6.1.2.37 HasContentElement

[SWS_DM_02364] Definition of API function ara::diag::DiagnosticSovdContentElementData::HasContentElementUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto HasContentElement (ara::core::StringView key) const noexcept -> ara::core::Result< bool >;	
Parameters (in):	key	The key
Return value:	ara::core::Result< bool >	boolean indicating if the elements is found or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData contains an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.38 HasError

[SWS_DM_02317] Definition of API function `ara::diag::DiagnosticSovdContentElementData::HasError`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto HasError () const noexcept -> ara::core::Result< void >;	
Return value:	ara::core::Result< void >	void result if it doesn't contain an error, otherwise the ara::diag::DiagSovdErrc
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the content contains an error.	

]

8.60.6.1.2.39 PushBack

[SWS_DM_02346] Definition of API function `ara::diag::DiagnosticSovdContentElementData::PushBack`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto PushBack (std::int64_t content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with signed int as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.6.1.2.40 PushBack

[SWS_DM_02347] Definition of API function `ara::diag::DiagnosticSovdContentElementData::PushBack`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto PushBack (std::uint64_t content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with unsigned int as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

8.60.6.1.2.41 PushBack

[SWS_DM_02348] Definition of API function `ara::diag::DiagnosticSovdContentElementData::PushBack`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<pre>auto PushBack (double content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with double as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

8.60.6.1.2.42 PushBack

[SWS_DM_02349] Definition of API function ara::diag::DiagnosticSovdContentElementData::PushBackUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto PushBack (bool content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with boolean as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.6.1.2.43 PushBack

[SWS_DM_02350] Definition of API function ara::diag::DiagnosticSovdContentElementData::PushBackUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto PushBack (ara::core::StringView content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData with string as primitive type. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.6.1.2.44 PushBack

[SWS_DM_02351] Definition of API function ara::diag::DiagnosticSovdContentElementData::PushBackUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto PushBack (DiagnosticSovdContentElementData content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	content_element	The content element to push back
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Pushes back a DiagnosticSovdContentElementData. The current element shall be of ArrayContentElementData and the element to add of the specified in the ArrayContentElementData constructor	

]

8.60.6.1.2.45 RecordBegin

[SWS_DM_02359] Definition of API function ara::diag::DiagnosticSovdContentElementData::RecordBeginUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto RecordBegin () noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataIterator >;	
Return value:	ara::core::Result< DiagnosticSovdRecordContentElementDataIterator >	The iterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.46 RecordCBegin

[SWS_DM_02361] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordCBegin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto RecordCBegin () noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator ></code>	The iterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	<code>DiagErrc::kInvalidArgument</code>	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.6.1.2.47 RecordCEnd

[SWS_DM_02362] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordCEnd`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto RecordCEnd () noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator >;</code>	
Return value:	<code>ara::core::Result< DiagnosticSovdRecordContentElementDataConstIterator ></code>	The iterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	<code>DiagErrc::kInvalidArgument</code>	rollback_semantics
		e.g. kPreFailed with internal debouncing





Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData

]

8.60.6.1.2.48 RecordCapacity

[SWS_DM_02354] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordCapacity`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto RecordCapacity () const noexcept -> ara::core::Result< std::size_t >;</code>	
Return value:	<code>ara::core::Result< std::size_t ></code>	The capacity of the RecordContentElementData or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.49 RecordCount

[SWS_DM_02353] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordCount`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto RecordCount () const noexcept -> ara::core::Result< std::size_t >;</code>	
Return value:	<code>ara::core::Result< std::size_t ></code>	The count of the RecordContentElementData or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	





Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns the count of elements in the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.50 RecordEmpty

[SWS_DM_02355] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordEmpty`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto RecordEmpty () const noexcept -> ara::core::Result< bool >;	
Return value:	ara::core::Result< bool >	If the record is empty or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData is empty. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.51 RecordEnd

[SWS_DM_02360] Definition of API function `ara::diag::DiagnosticSovdContentElementData::RecordEnd`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto RecordEnd () noexcept -> ara::core::Result< DiagnosticSovdRecordContentElementDataIterator >;	





Return value:	ara::core::Result<DiagnosticSovdRecordContentElementData Iterator >	The iterator or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.52 RecordRemoveAll

[SWS_DM_02356] Definition of API function ara::diag::DiagnosticSovdContentElementData::RecordRemoveAll

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto RecordRemoveAll () noexcept -> ara::core::Result< void >;	
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Removes all the content element in the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.53 RecordReserve

[SWS_DM_02363] Definition of API function ara::diag::DiagnosticSovdContentElementData::RecordReserve

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	





Syntax:	auto RecordReserve (std::size_t capacity) noexcept -> ara::core::Result< void >;	
Parameters (in):	capacity	The capacity
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reserves the capacity of the RecordContentElementData.	

]

8.60.6.1.2.54 Remove

[SWS_DM_02367] Definition of API function ara::diag::DiagnosticSovdContentElementData::Remove

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto Remove (ara::core::StringView key) noexcept -> ara::core::Result< bool >;	
Parameters (in):	key	The key
Return value:	ara::core::Result< bool >	A boolean indicating if the element could be removed or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Removes an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.6.1.2.55 Reserve

[SWS_DM_02345] Definition of API function ara::diag::DiagnosticSovdContentElementData::ReserveUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto Reserve (std::size_t capacity) noexcept -> ara::core::Result< void >;	
Parameters (in):	capacity	The capacity
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reserves the capacity of the DiagnosticSovdArrayContentElementData.	

]

8.60.6.1.2.56 SetBool

[SWS_DM_02324] Definition of API function ara::diag::DiagnosticSovdContentElementData::SetBoolUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto SetBool (bool value) -> ara::core::Result< void >;	
Parameters (in):	value	The value to set
Return value:	ara::core::Result< void >	Void result or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Set bool.	

]

8.60.6.1.2.57 SetDouble

[SWS_DM_02323] Definition of API function `ara::diag::DiagnosticSovdContentElementData::SetDouble`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>auto SetDouble (double value) -> ara::core::Result< void >;</code>	
Parameters (in):	value	The value to set
Return value:	<code>ara::core::Result< void ></code>	Void result or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Set double.	

8.60.6.1.2.58 SetError

[SWS_DM_02318] Definition of API function `ara::diag::DiagnosticSovdContentElementData::SetError`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementData</code>	
Syntax:	<code>void SetError (ara::core::ErrorCode error_code) noexcept;</code>	
Parameters (in):	error_code	The error to set
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Sets the content to be erroneous.	

8.60.6.1.2.59 SetSignedInt

[SWS_DM_02321] Definition of API function ara::diag::DiagnosticSovdContentElementData::SetSignedInt*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto SetSignedInt (std::int64_t value) -> ara::core::Result< void >;	
Parameters (in):	value	The value to set
Return value:	ara::core::Result< void >	Void result or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Set signed integer.	

]

8.60.6.1.2.60 SetString

[SWS_DM_02325] Definition of API function ara::diag::DiagnosticSovdContentElementData::SetString*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto SetString (ara::core::StringView value) -> ara::core::Result< void >;	
Parameters (in):	value	The value to set
Return value:	ara::core::Result< void >	Void result or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Set string.	

]

8.60.6.1.2.61 SetUnsignedInt

[SWS_DM_02322] Definition of API function ara::diag::DiagnosticSovdContentElementData::SetUnsignedInt

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto SetUnsignedInt (std::uint64_t value) -> ara::core::Result< void >;	
Parameters (in):	value	The value to set
Return value:	ara::core::Result< void >	Void result or error if DiagnosticSovdContentElementData is not of type PrimitiveContentElementData
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Set unsigned integer.	

]

8.60.6.1.2.62 Size

[SWS_DM_02335] Definition of API function ara::diag::DiagnosticSovdContentElementData::Size

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	auto Size () const noexcept -> ara::core::Result< std::size_t >;	
Return value:	ara::core::Result< std::size_t >	The size of the ArrayContentElementData or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the size of the Array content element. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

]

8.60.6.1.2.63 operator[]

[SWS_DM_02358] Definition of API function ara::diag::DiagnosticSovdContentElementData::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	<pre>auto operator[] (ara::core::StringView key) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > >;</pre>	
Parameters (in):	key	The key
Return value:	<pre>ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > ></pre>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.6.1.2.64 operator[]

[SWS_DM_02340] Definition of API function ara::diag::DiagnosticSovdContentElementData::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	<pre>auto operator[] (std::size_t index) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > >;</pre>	
DIRECTION NOT DEFINED	index	--
Return value:	<pre>ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > ></pre>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified index of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.6.1.2.65 operator[]

[SWS_DM_02357] Definition of API function ara::diag::DiagnosticSovdContentElementData::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	<pre>auto operator[] (ara::core::StringView key) noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > >;</pre>	
Parameters (in):	key	The key
Return value:	ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > >	reference to the content element or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.6.1.2.66 operator[]

[SWS_DM_02339] Definition of API function ara::diag::DiagnosticSovdContentElementData::operator[]

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementData	
Syntax:	<pre>auto operator[] (std::size_t index) noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > >;</pre>	
DIRECTION NOT DEFINED	index	--
Return value:	ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > >	reference to the content element or an error if DiagnosticSovdContentElementData is not of type ArrayContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified index of an array. This DiagnosticSovdContentElementData shall be of type ArrayContentElementData	

8.60.7 Class: DiagnosticSovdContentElementDataWithKey

[SWS_DM_02375] Definition of API class ara::diag::DiagnosticSovdContentElementDataWithKey

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DiagnosticSovdContentElementDataWithKey
Syntax:	<code>class DiagnosticSovdContentElementDataWithKey {...};</code>
Description:	A DiagnosticSovdContentElementData with associated key.

]

8.60.7.1 Public Member Functions

8.60.7.1.1 Special Member Functions

8.60.7.1.1.1 Destructor

[SWS_DM_02376] Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::~~DiagnosticSovdContentElementDataWithKey

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Scope:	<code>class ara::diag::DiagnosticSovdContentElementDataWithKey</code>
Syntax:	<code>~DiagnosticSovdContentElementDataWithKey () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Declare the Dtor and move/copy constructors/assignments .

]

8.60.7.1.2 Member Functions

8.60.7.1.2.1 GetContent

[SWS_DM_02379] Definition of API function `ara::diag::DiagnosticSovdContentElementDataWithKey::GetContent`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementDataWithKey</code>	
Syntax:	<code>auto GetContent () noexcept -> DiagnosticSovdContentElementData &;</code>	
Return value:	DiagnosticSovdContentElementData	Reference to the associated content
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Gets the associated content.	

]

8.60.7.1.2.2 GetContent

[SWS_DM_02380] Definition of API function `ara::diag::DiagnosticSovdContentElementDataWithKey::GetContent`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdContentElementDataWithKey</code>	
Syntax:	<code>auto GetContent () const noexcept -> DiagnosticSovdContentElementData const &;</code>	
Return value:	DiagnosticSovdContentElementData const	Reference to the associated content
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Gets the associated content.	

]

8.60.7.1.2.3 GetName

[SWS_DM_02377] Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::GetNameUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementDataWithKey	
Syntax:	auto GetName () const noexcept -> ara::core::StringView;	
Return value:	ara::core::StringView	The name of the key
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Gets the name of the key.	

]

8.60.7.1.2.4 SetContent

[SWS_DM_02381] Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::SetContentUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementDataWithKey	
Syntax:	auto SetContent (DiagnosticSovdContentElementData content) noexcept -> ara::core::Result< void >;	
Parameters (in):	content	The content to set
Return value:	ara::core::Result< void >	void result or error if the content is of type ContentData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Sets the associated content.	

]

8.60.7.1.2.5 SetName

[SWS_DM_02378] Definition of API function `ara::diag::DiagnosticSovdContentElementDataWithKey::SetName`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdContentElementDataWithKey	
Syntax:	void SetName (ara::core::StringView name) noexcept;	
Parameters (in):	name	The name of the key
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Sets the name of the key.	

]

8.60.8 Class: DiagnosticSovdRecordContentElementDataConstIterator

[SWS_DM_02264] Definition of API class `ara::diag::DiagnosticSovdRecordContentElementDataConstIterator`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DiagnosticSovdRecordContentElementDataConstIterator
Syntax:	<code>class DiagnosticSovdRecordContentElementDataConstIterator {...};</code>
Description:	Const iterator for a DiagnosticSovdRecordContentElementData.

]

8.60.8.1 Public Member Functions

8.60.8.1.1 Member Functions

8.60.8.1.1.1 operator!=

[SWS_DM_02268] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator!=`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator</code>	
Syntax:	<code>auto operator!= (DiagnosticSovdRecordContentElementDataConstIterator const &other) noexcept -> bool;</code>	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Not equal operator.	

]

8.60.8.1.1.2 operator*

[SWS_DM_02265] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator*`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator</code>	
Syntax:	<code>auto operator* () noexcept -> DiagnosticSovdContentElementDataWithKey const &;</code>	
Return value:	DiagnosticSovdContentElementDataWithKey const	Return a view on the content element
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.8.1.1.3 operator++

[SWS_DM_02269] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator++

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator</code>	
Syntax:	<code>auto operator++ () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator &;</code>	
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	Reference to the incremented iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Increment operator.	

]

8.60.8.1.1.4 operator->

[SWS_DM_02266] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator->

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator</code>	
Syntax:	<code>auto operator-> () noexcept -> DiagnosticSovdContentElementDataWithKey const *;</code>	
Return value:	DiagnosticSovdContentElementDataWithKey const *	Return a pointer to the DiagnosticSovdAbstractContentDataWithKey
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.8.1.1.5 operator==

[SWS_DM_02267] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator==Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator	
Syntax:	auto operator== (DiagnosticSovdRecordContentElementDataConstIterator const &other) noexcept -> bool;	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Equal operator.	

]

8.60.9 Class: DiagnosticSovdRecordContentElementDataConstView

[SWS_DM_02295] Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataConstViewUpstream requirements: [RS_Diag_04260](#)

[

Kind:	class	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdRecordContentElementDataConstView	
Syntax:	class DiagnosticSovdRecordContentElementDataConstView {...};	
Description:	Const view on a DiagnosticSovdContentElementData with Record type.	

]

8.60.9.1 Public Member Functions

8.60.9.1.1 Member Functions

8.60.9.1.1.1 Begin

[SWS_DM_02301] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstView::Begin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto Begin () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdRecord ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.9.1.1.2 CBegin

[SWS_DM_02303] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstView::CBegin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto CBegin () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdRecord ContentElementData ConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.9.1.1.3 CEnd

[SWS_DM_02304] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::CEnd*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto CEnd () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.9.1.1.4 Capacity

[SWS_DM_02298] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::Capacity*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto Capacity () const noexcept -> std::size_t;</code>	
Return value:	std::size_t	The capacity of the RecordContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.9.1.1.5 Empty

[SWS_DM_02299] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::Empty*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataConstView	
Syntax:	auto Empty () const noexcept -> bool;	
Return value:	bool	If the record is empty
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData is empty. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.9.1.1.6 End

[SWS_DM_02302] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::End*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataConstView	
Syntax:	auto End () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator ;	
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.9.1.1.7 Find

[SWS_DM_02306] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstView::Find`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto Find (ara::core::StringView key) const noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Parameters (in):	key	The key
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	A DiagnosticSovdRecordContentElementDataIterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Find an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.9.1.1.8 HasContentElement

[SWS_DM_02305] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataConstView::HasContentElement`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<code>auto HasContentElement (ara::core::StringView key) const noexcept -> ara::core::Result< bool >;</code>	
Parameters (in):	key	The key
Return value:	<code>ara::core::Result< bool ></code>	boolean indicating if the elements is found or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData contains an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.9.1.1.9 HasError

[SWS_DM_02296] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::HasError*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataConstView	
Syntax:	auto HasError () const noexcept -> ara::core::Result< void >;	
Return value:	ara::core::Result< void >	void result if it doesn't contain an error, otherwise the ara::diag::DiagSovdErrc
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the content contains an error.	

]

8.60.9.1.1.10 Size

[SWS_DM_02297] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::Size*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataConstView	
Syntax:	auto Size () const noexcept -> std::size_t;	
Return value:	std::size_t	The count of the RecordContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the count of elements in the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.9.1.1.11 operator[]

[SWS_DM_02300] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::operator[]Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataConstView</code>	
Syntax:	<pre>auto operator[] (ara::core::StringView key) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > >;</pre>	
Parameters (in):	key	The key
Return value:	<pre>ara::core::Result< std::reference_wrapper< DiagnosticSovdContent ElementData const > ></pre>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.10 Class: DiagnosticSovdRecordContentElementDataIterator

[SWS_DM_02258] Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataIteratorUpstream requirements: [RS_Diag_04260](#)

[

Kind:	class
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	DiagnosticSovdRecordContentElementDataIterator
Syntax:	<code>class DiagnosticSovdRecordContentElementDataIterator {...};</code>
Description:	Iterator for a DiagnosticSovdRecordContentElementData.

]

8.60.10.1 Public Member Functions

8.60.10.1.1 Member Functions

8.60.10.1.1.1 operator!=

[SWS_DM_02262] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator!=`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataIterator</code>	
Syntax:	<code>auto operator!= (DiagnosticSovdRecordContentElementDataIterator const &other) noexcept -> bool;</code>	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Not equal operator.	

]

8.60.10.1.1.2 operator*

[SWS_DM_02259] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator*`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataIterator</code>	
Syntax:	<code>auto operator* () noexcept -> DiagnosticSovdContentElementDataWithKey &;</code>	
Return value:	DiagnosticSovdContentElementDataWithKey	Return a view on the content element
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.10.1.1.3 operator++

[SWS_DM_02263] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator++Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataIterator	
Syntax:	auto operator++ () noexcept -> DiagnosticSovdRecordContentElementDataIterator &;	
Return value:	DiagnosticSovdRecordContentElementDataIterator	Reference to the incremented iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Increment operator.	

]

8.60.10.1.1.4 operator->

[SWS_DM_02260] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator->Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataIterator	
Syntax:	auto operator-> () noexcept -> DiagnosticSovdContentElementDataWithKey * ;	
Return value:	DiagnosticSovdContentElementDataWithKey *	Return a pointer to the DiagnosticSovdAbstractContentDataWithKey
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Dereference operator.	

]

8.60.10.1.1.5 operator==

[SWS_DM_02261] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator==*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataIterator	
Syntax:	auto operator== (DiagnosticSovdRecordContentElementDataIterator const &other) noexcept -> bool;	
Parameters (in):	other	The other iterator to test equality with
Return value:	bool	True if operator are equals, false otherwise
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Equal operator.	

]

8.60.11 Class: DiagnosticSovdRecordContentElementDataView

[SWS_DM_02270] Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataView*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	class	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	DiagnosticSovdRecordContentElementDataView	
Syntax:	class DiagnosticSovdRecordContentElementDataView {...};	
Description:	View on a DiagnosticSovdContentElementData with Record type.	

]

8.60.11.1 Public Member Functions

8.60.11.1.1 Member Functions

8.60.11.1.1.1 Add

[SWS_DM_02293] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, DiagnosticSovdContentElementData content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

8.60.11.1.1.2 Add

[SWS_DM_02292] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, ara::core::StringView content_ element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	





Description:	Adds an element with the specified key with string as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData

]

8.60.11.1.1.3 Add

[SWS_DM_02290] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Add`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	auto Add (ara::core::StringView key, double content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with double as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.11.1.1.4 Add

[SWS_DM_02291] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Add`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	auto Add (ara::core::StringView key, bool content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	key	The key
	content_element	The content element to add





Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with boolean as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.11.1.1.5 Add

[SWS_DM_02289] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Add

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Add (ara::core::StringView key, std::uint64_t content_element) noexcept -> ara::core::Result< void >;	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with unsigned integer as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

]

8.60.11.1.1.6 Add

[SWS_DM_02288] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Add`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<pre>auto Add (ara::core::StringView key, std::int64_t content_element) noexcept -> ara::core::Result< void >;</pre>	
Parameters (in):	key	The key
	content_element	The content element to add
Return value:	<code>ara::core::Result< void ></code>	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element cannot be added
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Adds an element with the specified key with signed integer as primitive value. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData and the element to add shall not be of type ContentData	

8.60.11.1.1.7 Begin

[SWS_DM_02279] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Begin`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<pre>auto Begin () noexcept -> DiagnosticSovdRecordContentElementData Iterator;</pre>	
Return value:	DiagnosticSovdRecordContentElementData Iterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.11.1.1.8 CBegin

[SWS_DM_02281] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::CBegin`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto CBegin () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the begin operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.9 CEnd

[SWS_DM_02282] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::CEnd`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto CEnd () noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.10 Capacity

[SWS_DM_02274] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Capacity`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Capacity () const noexcept -> std::size_t;	
Return value:	std::size_t	The capacity of the RecordContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the capacity of the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.11 Clear

[SWS_DM_02276] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Clear`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Clear () noexcept -> void;	
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Removes all the content element in the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.12 Empty

[SWS_DM_02275] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Empty`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto Empty () const noexcept -> bool;</code>	
Return value:	bool	If the record is empty
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData is empty. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.13 End

[SWS_DM_02280] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::End`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto End () noexcept -> DiagnosticSovdRecordContentElementData Iterator;</code>	
Return value:	DiagnosticSovdRecordContentElementData Iterator	The iterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the end operator of a RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.14 Erase

[SWS_DM_02294] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::EraseUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Erase (DiagnosticSovdRecordContentElementDataIterator iterator) noexcept -> ara::core::Result< void >;	
Parameters (in):	iterator	The iterator pointing to the element to erase
Return value:	ara::core::Result< void >	void result or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the element couldn't be erased
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	DiagErrc::kInvalidArgument	rollback_semantics
		e.g. kPreFailed with internal debouncing
Description:	Erases a content element from DiagnosticSovdRecordContentElementData. The current element shall be of RecordContentElementData or ContentData	

]

8.60.11.1.1.15 Find

[SWS_DM_02285] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::FindUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Find (ara::core::StringView key) noexcept -> DiagnosticSovdRecordContentElementDataIterator ;	
Parameters (in):	key	The key
Return value:	DiagnosticSovdRecordContentElementDataIterator	A DiagnosticSovdRecordContentElementDataIterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Find an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.16 Find

[SWS_DM_02286] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Find`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto Find (ara::core::StringView key) const noexcept -> DiagnosticSovdRecordContentElementDataConstIterator;</code>	
Parameters (in):	key	The key
Return value:	DiagnosticSovdRecordContentElementDataConstIterator	A DiagnosticSovdRecordContentElementDataIterator
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Find an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.11.1.1.17 HasContentElement

[SWS_DM_02284] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::HasContentElement`

Upstream requirements: [RS_Diag_04260](#)

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto HasContentElement (ara::core::StringView key) const noexcept -> ara::core::Result< bool >;</code>	
Parameters (in):	key	The key
Return value:	<code>ara::core::Result< bool ></code>	boolean indicating if the elements is found or an error if DiagnosticSovdAbstractContentData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the RecordContentElementData contains an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

8.60.11.1.1.18 HasError

[SWS_DM_02271] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::HasError*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto HasError () const noexcept -> ara::core::Result< void >;	
Return value:	ara::core::Result< void >	void result if it doesn't contain an error, otherwise the ara::diag::DiagSovdErrc
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns if the content contains an error.	

]

8.60.11.1.1.19 Remove

[SWS_DM_02287] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Remove*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Remove (ara::core::StringView key) noexcept -> ara::core::Result< bool >;	
Parameters (in):	key	The key
Return value:	ara::core::Result< bool >	A boolean indicating if the element could be removed or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Removes an element with the specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.20 Reserve

[SWS_DM_02283] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::Reserve`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Reserve (std::size_t capacity) noexcept -> void;	
Parameters (in):	capacity	The capacity
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Reserves the capacity of the RecordContentElementData.	

]

8.60.11.1.1.21 SetError

[SWS_DM_02272] Definition of API function `ara::diag::DiagnosticSovdRecordContentElementDataView::SetError`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	void SetError (ara::core::ErrorCode error_code) noexcept;	
Parameters (in):	error_code	The error to set
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Sets the content to be erroneous.	

]

8.60.11.1.1.22 Size

[SWS_DM_02273] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Size*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto Size () const noexcept -> std::size_t;	
Return value:	std::size_t	The count of the RecordContentElementData
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the count of elements in the RecordContentElementData. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.23 operator[]

[SWS_DM_02278] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::operator[]*Upstream requirements:* [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	class ara::diag::DiagnosticSovdRecordContentElementDataView	
Syntax:	auto operator[] (ara::core::StringView key) const noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData const > >;	
Parameters (in):	key	The key
Return value:	ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData const > >	reference to the content element or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.60.11.1.1.24 operator[]

[SWS_DM_02277] Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::operator[]

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/diagnostic_sovd_content_element_data.h"	
Scope:	<code>class ara::diag::DiagnosticSovdRecordContentElementDataView</code>	
Syntax:	<code>auto operator[] (ara::core::StringView key) noexcept -> ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > >;</code>	
Parameters (in):	key	The key
Return value:	<code>ara::core::Result< std::reference_wrapper< DiagnosticSovdContentElementData > ></code>	reference to the content element or an error if DiagnosticSovdContentElementData is not of type RecordContentElementData or ContentData OR the key is not found
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Access the content element with specified key. This DiagnosticSovdContentElementData shall be of type RecordContentElementData or ContentData	

]

8.61 Header: ara/diag/generic_sovd_content.h

8.61.1 Class: GenericDiagnosticSovdContent

[SWS_DM_02186] Definition of API class ara::diag::GenericDiagnosticSovdContent

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	class
Header file:	#include "ara/diag/generic_sovd_content.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	namespace ara::diag
Symbol:	GenericDiagnosticSovdContent
Syntax:	<code>class GenericDiagnosticSovdContent {...};</code>
Description:	Generic Diagnostic SOVD Content interface.

]

8.61.1.1 Public Member Functions

8.61.1.1.1 Special Member Functions

8.61.1.1.1.1 Destructor

[SWS_DM_02188] Definition of API function `ara::diag::GenericDiagnosticSovdContent::~~GenericDiagnosticSovdContent`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>
Syntax:	<code>virtual ~GenericDiagnosticSovdContent () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Destructor.

]

8.61.1.1.2 Constructors

8.61.1.1.2.1 GenericDiagnosticSovdContent

[SWS_DM_02187] Definition of API function `ara::diag::GenericDiagnosticSovdContent::GenericDiagnosticSovdContent`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>	
Syntax:	<code>explicit GenericDiagnosticSovdContent (ara::core::InstanceSpecifier const &specifier, SovdContentConcurrencyType concurrency_type) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticData IdentifierGenericInterface
	concurrency_type	The concurrency type
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructor for GenericDiagnosticSovdContent (inherited)	

]

8.61.1.1.3 Member Functions

8.61.1.1.3.1 Offer

[SWS_DM_02193] Definition of API function `ara::diag::GenericDiagnosticSovdContent::Offer`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Positive result if service was offered, error if offer failed
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	<code>DiagOfferErrc::kAlready Offered</code>	rollback_semantics
		The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

]

8.61.1.1.3.2 Read

[SWS_DM_02191] Definition of API function `ara::diag::GenericDiagnosticSovdContent::Read`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>	
Syntax:	<pre>virtual ara::core::Future< DiagnosticSovdContentData > Read (ara::core::StringView sovd_content_name, ara::diag::MetaInfo const &meta_info, ara::diag::CancellationHandler cancellation_handler) noexcept=0;</pre>	
Parameters (in):	<code>sovd_content_name</code>	The SOVD content name
	<code>meta_info</code>	contains additional meta information
	<code>cancellation_handler</code>	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.
Return value:	<code>ara::core::Future< DiagnosticSovdContent Data ></code>	A future with either SOVD abstract content data (for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Called for "Read Single Data Value request" for the specified SOVD Content.	

]

8.61.1.1.3.3 StopOffer

[SWS_DM_02194] Definition of API function ara::diag::GenericDiagnosticSovdContent::StopOfferUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"
Scope:	class ara::diag::GenericDiagnosticSovdContent
Syntax:	void StopOffer () noexcept;
Return value:	None
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.61.1.1.3.4 Write

[SWS_DM_02192] Definition of API function ara::diag::GenericDiagnosticSovdContent::WriteUpstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	class ara::diag::GenericDiagnosticSovdContent	
Syntax:	virtual ara::core::Future< void > Write (ara::core::StringView sovd_content_name, DiagnosticSovdContentData const &sovd_abstract_content_data, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept=0;	
Parameters (in):	sovd_content_name	The SOVD content name
	sovd_abstract_content_data	The SOVD abstract content data. Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler
	meta_info	contains additional meta information
	cancellation_handler	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.
Return value:	ara::core::Future< void >	A future with either void (for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Called for "Write a Data Value" request for the specified SOVD Content.	

]

8.61.1.2 Protected Member Functions

8.61.1.2.1 Special Member Functions

8.61.1.2.1.1 Move Constructor

[SWS_DM_02189] Definition of API function `ara::diag::GenericDiagnosticSovdContent::GenericDiagnosticSovdContent`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>	
Syntax:	<code>GenericDiagnosticSovdContent (GenericDiagnosticSovdContent &&other) noexcept;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructs an instance of <code>GenericDiagnosticSovdContent</code> .	
Visibility:	protected	

]

8.61.1.2.2 Member Functions

8.61.1.2.2.1 operator=

[SWS_DM_02190] Definition of API function `ara::diag::GenericDiagnosticSovdContent::operator=`

Upstream requirements: [RS_Diag_04260](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_content.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdContent</code>	
Syntax:	<code>auto operator= (GenericDiagnosticSovdContent &&other) & noexcept -> GenericDiagnosticSovdContent &;</code>	
Parameters (out):	other	The other object
Return value:	<code>GenericDiagnosticSovdContent</code>	Reference to self
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move assigns an instance of <code>GenericDiagnosticSovdContent</code> .	
Visibility:	protected	

]

8.62 Header: ara/diag/generic_diagnostic_sovd_operation.h

8.62.1 Non-Member Types

8.62.1.1 Enumeration: OperationExecutionCapability

[SWS_DM_02383] Definition of API enum ara::diag::OperationExecutionCapability

Upstream requirements: [RS_Diag_04262](#)

Kind:	enumeration	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	OperationExecutionCapability	
Underlying type:	std::uint8_t	
Syntax:	enum class OperationExecutionCapability : std::uint8_t {...};	
Values:	kExecute	The Operation can be executed.
	kStop	The Operation can get stopped.
	kStatus	The status of the Operation can be queried.
Description:	The standard operation execution capabilities .	

8.62.1.2 Enumeration: OperationExecutionStatus

[SWS_DM_02195] Definition of API enum ara::diag::OperationExecutionStatus

Upstream requirements: [RS_Diag_04262](#)

Kind:	enumeration	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	OperationExecutionStatus	
Underlying type:	std::uint8_t	
Syntax:	enum class OperationExecutionStatus : std::uint8_t {...};	
Values:	kRunning	The execution of an operation is running.
	kCompleted	The execution of an operation has completed successfully.
	kFailed	The execution of an operation has terminated with a fault.
	kStopped	The execution of an operation has been stopped by the client.
Description:	The operation execution status .	

8.62.1.3 Enumeration: TerminateOperationExecutionReason

[SWS_DM_02196] Definition of API enum ara::diag::TerminateOperationExecutionReason

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	enumeration	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	TerminateOperationExecutionReason	
Underlying type:	std::uint8_t	
Syntax:	enum class TerminateOperationExecutionReason : std::uint8_t {...};	
Values:	kClientRequest	The Client requests the operation to be stopped.
	kTimeout	The Execution shall be stopped due to timeout.
	kStateChange	The Execution shall be stopped due to a state change e.g. daemon restart, server is unlocked.
Description:	The Terminate Operation reason .	

]

8.62.2 Class: GenericDiagnosticSovdOperation

[SWS_DM_02203] Definition of API class ara::diag::GenericDiagnosticSovdOperation

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	class	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Forwarding header file:	#include "ara/diag/diag_fwd.h"	
Scope:	namespace ara::diag	
Symbol:	GenericDiagnosticSovdOperation	
Syntax:	class GenericDiagnosticSovdOperation {...};	
Description:	Generic Diagnostic SOVD Operation interface .	

]

8.62.2.1 Public Member Functions

8.62.2.1.1 Special Member Functions

8.62.2.1.1.1 Destructor

[SWS_DM_02205] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::~~GenericDiagnosticSovdOperation`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>
Syntax:	<code>virtual ~GenericDiagnosticSovdOperation () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Destructor.

]

8.62.2.1.2 Constructors

8.62.2.1.2.1 GenericDiagnosticSovdOperation

[SWS_DM_02204] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::GenericDiagnosticSovdOperation`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>	
Syntax:	<code>explicit GenericDiagnosticSovdOperation (ara::core::InstanceSpecifier const &specifier, ConcurrencyType concurrency_type) noexcept;</code>	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticData IdentifierGenericInterface
	concurrency_type	The concurrency type
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructor for GenericDiagnosticSovdOperation (inherited)	

]

8.62.2.1.3 Member Functions

8.62.2.1.3.1 Offer

[SWS_DM_02211] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::Offer`

Upstream requirements: [RS_Diag_04262](#)

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>	
Syntax:	<code>ara::core::Result< void > Offer () noexcept;</code>	
Return value:	<code>ara::core::Result< void ></code>	Returns nothing or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	<code>DiagOfferErrc::kAlready Offered</code>	rollback_semantics The service is already offered.
Description:	This Offer will enable the DM to forward request messages to this handler.	

8.62.2.1.3.2 StartExecution

[SWS_DM_02210] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::StartExecution`

Upstream requirements: [RS_Diag_04262](#)

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>	
Syntax:	<code>virtual ara::core::Future< ara::core::Variant< StartSynchronous ExecutionResponse, StartAsynchronousExecutionResponse > > Start Execution (ara::core::StringView sovd_operation_name, DiagnosticSovd ContentData const &sovd_content_data, ara::diag::MetaInfo const &meta_info, ara::diag::CancellationHandler cancellation_handler) noexcept=0;</code>	
Parameters (in):	<code>sovd_operation_name</code>	The SOVD operation name
	<code>sovd_content_data</code>	The SOVD content data. Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler
	<code>meta_info</code>	contains additional meta information
	<code>cancellation_handler</code>	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.





Return value:	ara::core::Future< ara::core::Variant< Start SynchronousExecution Response, Start AsynchronousExecution Response > >	A future with either StartSynchronousExecutionResponse or Start AsynchronousExecutionResponse(for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for "Start execution of an operation" for the specified SOVD Operation.	

]

8.62.2.1.3.3 StopOffer

[SWS_DM_02212] Definition of API function ara::diag::GenericDiagnosticSovd Operation::StopOffer

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>
Syntax:	<code>void StopOffer () noexcept;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	This StopOffer will disable the forwarding of request messages from DM.

]

8.62.2.2 Protected Member Functions

8.62.2.2.1 Special Member Functions

8.62.2.2.1.1 Move Constructor

[SWS_DM_02206] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::GenericDiagnosticSovdOperation`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>	
Syntax:	<code>GenericDiagnosticSovdOperation (GenericDiagnosticSovdOperation &&other) noexcept;</code>	
Parameters (out):	other	The other object
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move constructs an instance of <code>GenericDiagnosticSovdOperation</code> .	
Visibility:	protected	

]

8.62.2.2.2 Member Functions

8.62.2.2.2.1 `operator=`

[SWS_DM_02207] Definition of API function `ara::diag::GenericDiagnosticSovdOperation::operator=`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::GenericDiagnosticSovdOperation</code>	
Syntax:	<code>auto operator= (GenericDiagnosticSovdOperation &&other) & noexcept -> GenericDiagnosticSovdOperation &;</code>	
Parameters (out):	other	The other object
Return value:	<code>GenericDiagnosticSovdOperation</code>	Reference to self
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Move assigns an instance of <code>GenericDiagnosticSovdOperation</code> .	
Visibility:	protected	

]

8.62.3 Struct: StartAsynchronousExecutionResponse

[SWS_DM_02209] Definition of API class ara::diag::GenericDiagnosticSovdOperation::StartAsynchronousExecutionResponse

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	struct
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	class ara::diag::GenericDiagnosticSovdOperation
Symbol:	StartAsynchronousExecutionResponse
Syntax:	struct StartAsynchronousExecutionResponse {...};
Description:	Start Execution response in case of asynchronous operation .

]

8.62.4 Struct: StartSynchronousExecutionResponse

[SWS_DM_02208] Definition of API class ara::diag::GenericDiagnosticSovdOperation::StartSynchronousExecutionResponse

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	struct
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	class ara::diag::GenericDiagnosticSovdOperation
Symbol:	StartSynchronousExecutionResponse
Syntax:	struct StartSynchronousExecutionResponse {...};
Description:	Start Execution response in case of synchronous operation .

]

8.62.5 Class: SovdOperationExecution

[SWS_DM_02197] Definition of API class ara::diag::SovdOperationExecution

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	class
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"





Scope:	namespace ara::diag
Symbol:	SovdOperationExecution
Syntax:	<code>class SovdOperationExecution {...};</code>
Description:	Interface to handle the SOVD Operation Execution .

]

8.62.5.1 Public Member Functions

8.62.5.1.1 Special Member Functions

8.62.5.1.1.1 Default Constructor

[SWS_DM_02198] Definition of API function `ara::diag::SovdOperationExecution::SovdOperationExecution`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/generic_diagnostic_sovd_operation.h"</code>
Scope:	<code>class ara::diag::SovdOperationExecution</code>
Syntax:	<code>SovdOperationExecution () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Constructor of SovdOperationExecution (inherited) -.

]

8.62.5.1.1.2 Destructor

[SWS_DM_02199] Definition of API function `ara::diag::SovdOperationExecution::~~SovdOperationExecution`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function
Header file:	<code>#include "ara/diag/generic_diagnostic_sovd_operation.h"</code>
Scope:	<code>class ara::diag::SovdOperationExecution</code>
Syntax:	<code>virtual ~SovdOperationExecution () noexcept=default;</code>
Exception Safety:	exception safe
Description:	Destructor of SovdOperationExecution -.

]

8.62.5.2 Protected Member Functions

8.62.5.2.1 Member Functions

8.62.5.2.1.1 ExecuteCapability

[SWS_DM_02384] Definition of API function `ara::diag::SovdOperationExecution::ExecuteCapability`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::SovdOperationExecution</code>	
Syntax:	virtual ara::core::Future< <code>OperationExecutionStatus</code> > ExecuteCapability (<code>OperationExecutionCapability</code> capability, DiagnosticSovdContentData const &sovd_content_data, ara::diag::MetaInfo const &meta_info, <code>ara::diag::CancellationHandler</code> cancellation_handler) noexcept=0;	
Parameters (in):	capability	The SOVD operation execution capability
	sovd_content_data	The SOVD content data. Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler
	meta_info	contains additional meta information
	cancellation_handler	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.
Return value:	ara::core::Future< <code>OperationExecutionStatus</code> >	A future with either execution status (for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for " Support for execute / freeze / reset" for the specified SOVD Operation.	
Visibility:	protected	

]

8.62.5.2.1.2 GetStatus

[SWS_DM_02201] Definition of API function `ara::diag::SovdOperationExecution::GetStatus`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	<code>class ara::diag::SovdOperationExecution</code>	





Syntax:	virtual ara::core::Future< GetStatusResponse > GetStatus (ara::diag::MetaInfo const &meta_info, ara::diag::CancellationHandler cancellation_handler) noexcept=0;	
Parameters (in):	meta_info	contains additional meta information
	cancellation_handler	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.
Return value:	ara::core::Future< GetStatusResponse >	A future with either GetStatusResponse (for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for "Get status execution of an operation" for the current execution.	
Visibility:	protected	

]

8.62.5.2.1.3 Terminate

[SWS_DM_02202] Definition of API function ara::diag::SovdOperationExecution::Terminate

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	function	
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"	
Scope:	class ara::diag::SovdOperationExecution	
Syntax:	virtual ara::core::Future< void > Terminate (TerminateOperationExecutionReason reason, ara::diag::MetaInfo const &meta_info, ara::diag::CancellationHandler cancellation_handler) noexcept=0;	
Parameters (in):	reason	The terminate operation execution reason
	meta_info	contains additional meta information
	cancellation_handler	Informs if the current request is cancelled. This object shall not be stored or accessed beyond the lifetime of this operation.
Return value:	ara::core::Future< void >	A future with either void (for a positive response message) or a SOVD Error (for an negative response message)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	Please use an applicable SOVD error code, as defined in ISO 17978-3 / ASAM SOVD 1.1.	
Description:	Called for "Terminate execution of an operation" for the current execution.	
Visibility:	protected	

]

8.62.6 Struct: GetStatusResponse

[SWS_DM_02200] Definition of API class `ara::diag::SovdOperationExecution::GetStatusResponse`

Upstream requirements: [RS_Diag_04262](#)

[

Kind:	struct
Header file:	#include "ara/diag/generic_diagnostic_sovd_operation.h"
Forwarding header file:	#include "ara/diag/diag_fwd.h"
Scope:	<code>class ara::diag::SovdOperationExecution</code>
Symbol:	GetStatusResponse
Syntax:	<code>struct GetStatusResponse { ... };</code>
Description:	Get Status response .
Visibility:	protected

]

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 [23]. This chapter defines the default values for attributes and semantic constraints for elements specified in [23] that are part of the configuration model of this functional cluster.

The configuration is realized in huge parts using the Autosar Diagnostic Extract Template [4].

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [23].

10.2 Semantic Constraints

This section defines semantic constraints for elements specified in [23] that are part of the configuration model of this functional cluster.

[SWS_DM_CONSTR_00397] Configurable Namespace for Diagnostic Management [`DiagnosticPortInterface.namespace` may only exist for the following `DiagnosticPortInterfaces`:

- `DiagnosticRoutineInterface`
- `DiagnosticDataIdentifierInterface`
- `DiagnosticDataElementInterface`

]

[SWS_DM_CONSTR_02143] Mapping bus type to CPP Implementation data type for scalar types [

baseTypeEncoding	CPP implementation data type
BOOLEAN	uint8_t
NONE (For baseTypeSize = 8)	uint8_t
NONE (For baseTypeSize = 16)	uint16_t
NONE (For baseTypeSize = 32)	uint32_t
NONE (For baseTypeSize = 64)	uint64_t
2C (For baseTypeSize = 8)	int8_t
2C (For baseTypeSize = 16)	int16_t
2C (For baseTypeSize = 32)	int32_t
2C (For baseTypeSize = 64)	int64_t
BCD-P	uint8_t
BCD-UP	uint8_t
IEEE754 (For baseTypeSize = 32)	float
UTF-8	ara::core::String
For arrays with fixed-size it shall be implemented as <code>ara::core::Array</code> and for arrays with variable-size it shall be implemented as <code>ara::core::Vector</code> .	

]

[SWS_DM_CONSTR_02144] Mapping bus types to CPP Implementation data type for scalar types [In the case addressed by this spec item, it is only possible to define the [DiagnosticDataElement](#) as a scalar or an array (see [TPS_DEXT_01001], [TPS_DEXT_01002]). The ability to define a Variable-Size Array shall only be used for the last element of the [DID](#).]

A Mentioned Manifest Elements

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

This chapter is generated.

Class	ApApplicationError			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, Identifiable , MultilanguageReferrable , PackageableElement, 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			
Note	An argument of an operation, carries direction and implementation information.			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype , DataPrototype, Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	AtpClassifier.atpFeature, ClientServerOperation.argument			
Attribute	Type	Mult.	Kind	Note
direction	ArgumentDirectionEnum	0..1	attr	This attribute specifies the direction of the argument.
serverArgumentImplPolicy	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 <code>useArgumentType</code> for primitive arguments and structures.

Table A.2: ArgumentDataPrototype

Enumeration	ArgumentDirectionEnum
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.
Aggregated by	ArgumentDataPrototype.direction , DiagnosticSovdAccessArgument.direction, RunnableEntityArgument.direction, SwServiceArg.direction
Literal	Description
in	The argument value is passed to the callee. Tags: atp.EnumerationLiteralIndex=0





Enumeration	ArgumentDirectionEnum
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)			
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)			
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
baseTypeDefinition	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			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	ARObject, BaseTypeDefinition			
Aggregated by	BaseType.baseTypeDefinition			
Attribute	Type	Mult.	Kind	Note
baseTypeEncoding	BaseTypeEncodingString	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





Class	BaseTypeDirectDefinition			
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			
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 , DiagnosticExtendedDataRecordInterface.provide , DiagnosticRoutineInterface.requestResult , DiagnosticRoutineInterface.start , DiagnosticRoutineInterface.stop , PhmRecoveryActionInterface.recovery, ServiceInterface.method			
Attribute	Type	Mult.	Kind	Note
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). This Attribute is only used by the AUTOSAR Adaptive Platform.
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation. This Attribute is only used by the AUTOSAR Adaptive Platform.
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. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.7: ClientServerOperation

Class	CompuConst			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	<i>ARObject</i>			
Aggregated by	Compu.compuDefaultValue, CompuScale.compuInverseValue , CompuScaleConstantContents.compuConst			
Attribute	Type	Mult.	Kind	Note
compuConstContent 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			
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	<i>ARObject</i> , <i>CompuConstContent</i>			
Aggregated by	CompuConst.compuConstContentType			
Attribute	Type	Mult.	Kind	Note
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			
Note	This meta-class represents the textual content of a scale.			
Base	<i>ARObject</i> , <i>CompuConstContent</i>			
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			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	<i>ARObject</i>			
Aggregated by	CompuScales.compuScale			
Attribute	Type	Mult.	Kind	Note





Class	CompuScale			
a2IDisplayText	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 <code>CompuMethod</code> in A2L.
compuInverseValue	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
compuScaleContents	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	MultiLanguageOverviewParagraph	0..1	aggr	<desc> represents a general but brief description of the object in question. Tags: xml.sequenceOffset=30
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			
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





Class	CompuScaleConstantContents			
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			
Note	This meta-class represents the ability to stepwise express a computation method.			
Base	<i>ARObject</i> , <i>CompuContent</i>			
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	<i>CppImplementationDataType</i> (abstract)			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>AbstractImplementationDataType</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>AutosarDataType</i> , <i>CollectableElement</i> , <i>CppImplementationDataTypeContextTarget</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
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 CppImplementationDataType.
subElement (ordered)	CppImplementationDataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType





Class	CppImplementationDataType (abstract)			
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	CppImplementationDataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef).

Table A.14: CppImplementationDataType

Class	DiagEventDebounceCounterBased			
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 DemDebounceCounterBased.			
Base	ARObject, DiagEventDebounceAlgorithm , Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	DiagnosticDebounceAlgorithmProps.debounceAlgorithm , DiagnosticEventNeeds.diagEventDebounceAlgorithm			
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





Class	DiagEventDebounceCounterBased			
counterPassedThreshold	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			
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			
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)			
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





Class	DiagnosticAbstractDataIdentifier (abstract)			
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)			
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 DiagnosticRoutineSubfunction. Stereotypes: atpIdentityContributor Tags: atp.Status=candidate
dataElement	DiagnosticDataElement	0..1	aggr	This represents the related dataElement of the DiagnosticParameter Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=dataElement.shortName, dataElement.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			
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: atpSplittable Tags: atp.Splitkey=authenticationEnabled
diagnostic Session	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions Stereotypes: atpSplittable Tags: atp.Splitkey=diagnosticSession





Class	DiagnosticAccessPermission			
environmentalCondition	DiagnosticEnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission. Stereotypes: atpSplitable Tags: atp.Splitkey=environmentalCondition
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 This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.20: DiagnosticAccessPermission

Class	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	DiagnosticOperationCycle	0..1	ref	This represents the applicable aging cycle. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=agingCycle.diagnosticOperationCycle, agingCycle.variationPoint.shortLabel vh.latestBindingTime=postBuild
threshold	PositiveInteger	0..1	attr	Number of aging cycles needed to unlearn/delete the event. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild

Table A.21: DiagnosticAging

Class	DiagnosticAuthRole			
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			
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)			
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, DiagnosticVerifyCertificateUnidirectional			
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			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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			
Note	This mapping class identifies the PortPrototype in the application software that handles the client authentication. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement, Referrable			





Class	DiagnosticAuthenticationPortMapping			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
diagnosticAuthentication	DiagnosticAuthentication	0..1	ref	Reference to the DiagnosticAuthentication that is assigned to a SWC service port.
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticAuthenticationPortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atpSplittable Tags: atp.Splitkey=process

Table A.26: DiagnosticAuthenticationPortMapping

Class	DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. Tags: atp.recommendedPackage=DiagnosticConditions This Class is only used by the AUTOSAR Adaptive Platform.			
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			
Note	Defines to which SWC service ports the DiagnosticClearCondition is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
clearCondition	DiagnosticClearCondition	0..1	ref	Reference to the ClearCondition 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: atpSplittable Tags: atp.Splitkey=process
rPortPrototypeInExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticClearConditionMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.28: DiagnosticClearConditionPortMapping

Enumeration	DiagnosticClearDtcLimitationEnum
Note	Scope of the DEM_ClearDTC Api.
Aggregated by	DiagnosticMemoryDestination.clearDtcLimitation
Literal	Description
allSupportedDtc	DEM_ClearDtc API accepts all supported DTC values. Tags: atp.EnumerationLiteralIndex=0
clearAllDtc	DEM_ClearDtc API accepts ClearAllDTCs only. Tags: atp.EnumerationLiteralIndex=1

Table A.29: DiagnosticClearDtcLimitationEnum

Enumeration	DiagnosticClearEventAllowedBehaviorEnum
Note	This enumeration defines the possible behavior for clear event allowed
Aggregated by	DiagnosticEvent.clearEventAllowedBehavior
Literal	Description
noStatusByteChange	The event status byte keeps unchanged. Tags: atp.EnumerationLiteralIndex=0
onlyThisCycleAndReadiness	The OperationCycle and readiness bits of the event status byte are reset. Tags: atp.EnumerationLiteralIndex=1

Table A.30: DiagnosticClearEventAllowedBehaviorEnum

Class	DiagnosticComControl			
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
comControlClass	DiagnosticComControlClass	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.
customSubFunctionNumber	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			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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			
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.
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 This Attribute is only used by the AUTOSAR Adaptive Platform.
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 This Attribute is only used by the AUTOSAR Adaptive Platform.
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).





Class	«atpVariation» DiagnosticCommonProps			
responseOnSecondDeclinedRequest	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.
sovdProperties	DiagnosticSovdProps	1	aggr	This aggregation contains the definition of common properties that are specific for SOVD. This Attribute is only used by the AUTOSAR Adaptive Platform.
typeOfEventCombinationSupported	DiagnosticEventCombinationBehaviorEnum	0..1	attr	Select type of Event Combination support.

Table A.33: DiagnosticCommonProps

Class	<i>DiagnosticCondition</i> (abstract)			
Note	Abstract element for StorageConditions and EnableConditions.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , Identifiable , MultilanguageReferrable , <i>PackageableElement</i> , Referrable			
Subclasses	DiagnosticClearCondition , DiagnosticEnableCondition , <i>DiagnosticStorageCondition</i>			
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	<i>DiagnosticConditionInterface</i>			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , DiagnosticPortInterface , Identifiable , MultilanguageReferrable , <i>PackageableElement</i> , PortInterface , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.35: DiagnosticConditionInterface

Class	<i>DiagnosticConnectedIndicator</i>			
Note	Description of indicators that are defined per DiagnosticEvent.			
Base	<i>ARObject</i> , Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	DiagnosticEvent.connectedIndicator			





Class	DiagnosticConnectedIndicator			
Attribute	Type	Mult.	Kind	Note
behavior	DiagnosticConnectedIndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycleCounterThreshold	PositiveInteger	0..1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
indicator	DiagnosticIndicator	0..1	ref	Reference to the used indicator.
indicatorFailureCycleCounterThreshold	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			
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 DiagnosticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContributionSet. Tags: atp.recommendedPackage=DiagnosticContributionSets			
Base	ARElement, ARObject, CollectableElement, Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
commonProperties	DiagnosticCommonProps	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire DiagnosticContributionSet. Stereotypes: atpSplitable Tags: atp.Splitkey=commonProperties
element	DiagnosticCommonElement	*	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 DiagnosticContributionSet. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=serviceTable.diagnosticServiceTable, serviceTable.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table A.37: DiagnosticContributionSet

Class	DiagnosticControlDTCSetting			
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





Class	DiagnosticControlDTCSetting			
dtcSettingClass	DiagnosticControlDTCSettingClass	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			
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
customServiceClass	DiagnosticCustomServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustomServiceClass.

Table A.39: DiagnosticCustomServiceInstance

Class	DiagnosticDTCInformationInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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)			
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, DiagnosticWriteDataByIdentifier			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataIdentifier	DiagnosticAbstractDataIdentifier	0..1	ref	This represents the linked DiagnosticDataIdentifier.

Table A.41: DiagnosticDataByIdentifier

Class	DiagnosticDataElement			
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
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOfElements	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 DiagnosticDataElement if used with DiagnosticReadScalingDataByIdentifier
swDataDefProps	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			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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			
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 , DiagnosticCommonElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the DiagnosticDataIdentifier. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=dataElement.bitOffset, dataElement.ident.shortName, dataElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the DiagnosticDataIdentifier.





Class	DiagnosticDataIdentifier			
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfo Byte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table A.44: DiagnosticDataIdentifier

Class	DiagnosticDataIdentifierGenericInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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			
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





Class	DiagnosticDataIdentifierSet			
diagnosticData Identifier (ordered)	DiagnosticDataIdentifier	*	ref	Reference to an ordered list of Data Identifiers. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=diagnosticDataIdentifier.diagnosticData Identifier, diagnosticDataIdentifier.variationPoint.short Label vh.latestBindingTime=postBuild

Table A.47: DiagnosticDataIdentifierSet

Class	DiagnosticDataPortMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>DiagnosticMapping</i> , <i>DiagnosticSwMapping</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
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 Identifier.
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: atpSplittable Tags: atp.Splitkey=process

Table A.48: DiagnosticDataPortMapping

Class	DiagnosticDebounceAlgorithmProps			
Note	Defines properties for the debounce algorithm class.			
Base	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
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			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticDoIPActivationLineInterface			
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			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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	DiagnosticDoIPPowerModelInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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: DiagnosticDoIPPowerModelInterface

Class	DiagnosticDoIPTriggerVehicleAnnouncementInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.54: DiagnosticDoIPTriggerVehicleAnnouncementInterface

Class	DiagnosticDoIPActivationLinePortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP activation line. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractDoIPPortMapping, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
network Configuration Design	DoIPNetwork ConfigurationDesign	0..1	ref	This reference identifies the applicable DoIPNetwork ConfigurationDesign.

Table A.55: DiagnosticDoIPActivationLinePortMapping

Class	DiagnosticDoIPEntityIdentificationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP identity identification. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractDoIPPortMapping, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.56: DiagnosticDoIPEntityIdentificationPortMapping

Class	DiagnosticDoIPGroupIdentificationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP group identification. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractDoIPPortMapping, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.57: DiagnosticDoIPGroupIdentificationPortMapping

Class	DiagnosticDolpPowerModePortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the DoIP power mode. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AbstractDolpPortMapping, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.58: DiagnosticDolpPowerModePortMapping

Class	DiagnosticDownloadInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.59: DiagnosticDownloadInterface

Class	DiagnosticDynamicallyDefineDataIdentifier			
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	DiagnosticDynamicData Identifier	0..1	ref	This represents the applicable DiagnosticDynamicData Identifier.
dynamically DefineData IdentifierClass	DiagnosticDynamically DefineDataIdentifier 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 DiagnosticDynamicallyDefine DataIdentifier in the given context.
maxSource Element	PositiveInteger	0..1	attr	This represents the maximum number of source elements of the dynamically created DID.

Table A.60: DiagnosticDynamicallyDefineDataIdentifier

Class	DiagnosticDynamicallyDefineDataIdentifierClass			
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			





Class	DiagnosticDynamicallyDefineDataIdentifierClass			
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 0xF200 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.
configurationHandling	DiagnosticHandleDDIDConfigurationEnum	0..1	attr	This configuration switch defines whether DDID definition is handled as non-volatile information or not.
subfunction	DiagnosticDynamicallyDefineDataIdentifierSubfunctionEnum	*	attr	This attribute contains a list of applicable subfunctions for all DiagnosticDynamicallyDefineDataIdentifier that reference the DiagnosticDynamicallyDefineDataIdentifier Class.

Table A.61: DiagnosticDynamicallyDefineDataIdentifierClass

Class	DiagnosticEcuReset			
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
customSubFunctionNumber	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	DiagnosticEcuResetClass	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.62: DiagnosticEcuReset

Class	DiagnosticEcuResetClass			
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
respondToReset	DiagnosticResponseToEcuResetEnum	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.63: DiagnosticEcuResetClass

Class	DiagnosticEcuResetInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.64: DiagnosticEcuResetInterface

Class	DiagnosticEdrDataProviderMapping			
Note	This meta-class represents the definition of a data provider for the content of an extended data record. Tags: atp.recommendedPackage=DiagnosticDataProviders			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
dataProvider	NameToken	0..1	attr	This attribute defines the kind of data that is associated with the referenced DiagnosticExtendedDataRecord Element.
extendedData RecordElement	DiagnosticExtended DataRecordElement	0..1	ref	This reference identifies the DiagnosticExtendedData RecordElement to which the data provider is associated with.

Table A.65: DiagnosticEdrDataProviderMapping

Class	DiagnosticEnableCondition			
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.66: DiagnosticEnableCondition

Class	DiagnosticEnableConditionPortMapping			
Note	Defines to which SWC service ports the DiagnosticEnableCondition 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
enableCondition	DiagnosticEnable Condition	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port.





Class	DiagnosticEnableConditionPortMapping			
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 This Attribute is only used by the AUTOSAR Adaptive Platform.
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: RPortPrototypeInExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.67: DiagnosticEnableConditionPortMapping

Class	DiagnosticEnvCompareCondition (abstract)			
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 , DiagnosticEnvSovdDataCondition			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	Type	Mult.	Kind	Note
compareType	DiagnosticCompareTypeEnum	0..1	attr	This attributes represents the concrete type of the comparison.

Table A.68: DiagnosticEnvCompareCondition

Class	DiagnosticEnvConditionFormula			
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 DiagnosticEnvConditionFormula.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	DiagnosticLogicalOperatorEnum	0..1	attr	This attribute represents the concrete operator (supported operators: and, or) of the condition formula.
part (ordered)	DiagnosticEnvConditionFormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

Table A.69: DiagnosticEnvConditionFormula

Class	DiagnosticEnvConditionFormulaPart (abstract)			
Note	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompareCondition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
Base	ARObject			
Subclasses	DiagnosticEnvCompareCondition , DiagnosticEnvConditionFormula			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.70: DiagnosticEnvConditionFormulaPart

Class	DiagnosticEnvDataCondition			
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.71: DiagnosticEnvDataCondition

Class	DiagnosticEnvDataElementCondition			
Note	This meta-class represents the ability to formulate a diagnostic environment condition based on the value of a data element owned by the application software.			
Base	ARObject, DiagnosticEnvCompareCondition , DiagnosticEnvConditionFormulaPart			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	Type	Mult.	Kind	Note
compareValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of the compare value against which the value taken from the application software shall be compared.
dataPrototype	DataPrototype	0..1	iref	This instanceRef represent the ability to access a data element owned by the application software on the AUTOSAR classic platform. InstanceRef implemented by: DataPrototypeInSystem InstanceRef
pPortPrototype	PPortPrototype	0..1	iref	This instanceRef identifies the PortPrototype from which the relevant information for the environment condition can be obtained. This InstanceRef is only relevant for the adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.
process	ProcessDesign	0..1	ref	This reference identifies the applicable ProcessDesign. This Attribute is only used by the AUTOSAR Adaptive Platform.





Class	DiagnosticEnvDataElementCondition			
swDataDef Props	SwDataDefProps	0..1	aggr	Via this aggregation it is possible to describe the properties of the data that is obtained from the application for the environmental condition. Stereotypes: atp.Splittable Tags: atp.Splitkey=swDataDefProps

Table A.72: DiagnosticEnvDataElementCondition

Class	DiagnosticEnvSovdDataCondition			
Note	A DiagnosticEnvSovdDataCondition is an atomic condition that compares the current value of the referenced DiagnosticSovdContentElement with a constant value defined by the ValueSpecification. All compareTypes are supported. Tags: atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
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. Tags: atp.Status=candidate
contentElement	DiagnosticSovdContentElement	0..1	ref	This reference represents the related diagnostic SOVD content element. Tags: atp.Status=candidate

Table A.73: DiagnosticEnvSovdDataCondition

Class	DiagnosticEnvironmentalCondition			
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 , Multilanguage , Referrable , 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.74: DiagnosticEnvironmentalCondition

Class	DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. Tags: atp.recommendedPackage=DiagnosticEvents			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable , Multilanguage , Referrable , PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note





Class	DiagnosticEvent			
associated Event Identification	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.
clearEvent Allowed Behavior	DiagnosticClearEvent AllowedBehaviorEnum	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
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=postBuild
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=connectedIndicator.shortName, connectedIndicator.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 false: 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 ClearPrestoredFreezeFrame 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.75: DiagnosticEvent

Enumeration	DiagnosticEventCombinationBehaviorEnum
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.76: DiagnosticEventCombinationBehaviorEnum

Enumeration	DiagnosticEventCombinationReportingBehaviorEnum
Note	Select reporting format of events. Applicable only for Event Combination on Retrieval.
Aggregated by	DiagnosticCommonProps.eventCombinationReportingBehavior
Literal	Description
reportingInChronologicalOrderOldestFirst	The reporting order for event combination on retrieval is the chronological storage order of the events Tags: atp.EnumerationLiteralIndex=1

Table A.77: DiagnosticEventCombinationReportingBehaviorEnum

Enumeration	DiagnosticEventDisplacementStrategyEnum
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.78: DiagnosticEventDisplacementStrategyEnum

Class	DiagnosticEventInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.79: DiagnosticEventInterface

Class	DiagnosticEventPortMapping			
Note	Defines to which SWC service ports the DiagnosticEvent 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.





Class	DiagnosticEventPortMapping			
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 This Attribute is only used by the AUTOSAR Adaptive Platform.
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: RPortPrototypeInExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.80: DiagnosticEventPortMapping

Class	DiagnosticEventToDebounceAlgorithmMapping			
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.81: DiagnosticEventToDebounceAlgorithmMapping

Class	DiagnosticEventToEnableConditionGroupMapping			
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.82: DiagnosticEventToEnableConditionGroupMapping

Class	DiagnosticEventToOperationCycleMapping			
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			





Class	DiagnosticEventToOperationCycleMapping			
Attribute	Type	Mult.	Kind	Note
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which an Operation Cycle is assigned.
diagnosticOperationCycle	DiagnosticOperationCycle	0..1	ref	Reference to an OperationCycle assigned to a Diagnostic Event. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=diagnosticOperationCycle.diagnosticOperationCycle, diagnosticOperationCycle.variation Point.shortLabel vh.latestBindingTime=postBuild

Table A.83: DiagnosticEventToOperationCycleMapping

Class	DiagnosticEventToTroubleCodeUdsMapping			
Note	Defines which UDS Diagnostic Trouble Code is applicable for a DiagnosticEvent. Tags: atp.recommendedPackage=DiagnosticMappings			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , DiagnosticMapping , Identifiable , MultilanguageReferrable , <i>PackageableElement</i> , 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	DiagnosticTroubleCodeUds	0..1	ref	Reference to an UDS Diagnostic Trouble Code assigned to a DiagnosticEvent.

Table A.84: DiagnosticEventToTroubleCodeUdsMapping

Class	DiagnosticEventWindow			
Note	This represents the ability to define the characteristics of the applicable event window			
Base	<i>ARObject</i>			
Aggregated by	DiagnosticResponseOnEvent.eventWindow			
Attribute	Type	Mult.	Kind	Note
eventWindowTime	DiagnosticEventWindowTimeEnum	0..1	attr	This attribute clarifies the validity of the eventWindow

Table A.85: DiagnosticEventWindow

Class	DiagnosticExtendedDataRecord			
Note	Description of an extended data record. Tags: atp.recommendedPackage=DiagnosticExtendedDataRecords			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , Identifiable , MultilanguageReferrable , <i>PackageableElement</i> , 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.
element (ordered)	DiagnosticExtendedDataRecordElement	*	aggr	This aggregation represents the collection of elements of the enclosing Extended Data Record.





Class	DiagnosticExtendedDataRecord			
recordNumber	PositiveInteger	0..1	attr	This attribute specifies an unique identifier for an extended data record.
trigger	DiagnosticRecordTriggerEnum	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.86: DiagnosticExtendedDataRecord

Class	DiagnosticExtendedDataRecordElement			
Note	This meta-class represents an element of an extended data record. Such an element represents a primitive data object or a fixed-size array of primitive data objects without further internal structure (no structure allowed).			
Base	<i>ARObject</i> , Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	DiagnosticExtendedDataRecord.element			
Attribute	Type	Mult.	Kind	Note
numberOfElements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into a fixed-size array of data. The attribute determines the size of the array in terms of how many elements the array shall take. The aggregation of SwDataDefProps (especially with respect to SwBaseType) applies specifically to an element of the array (as a representation of all array elements) instead of the entire array itself!
swDataDefProps	SwDataDefProps	0..1	aggr	This aggregation allows for the specification of various properties for the enclosing element.

Table A.87: DiagnosticExtendedDataRecordElement

Class	DiagnosticExtendedDataRecordInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for providing values to an extended data record from within an application. Tags: atp.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
provide	ClientServerOperation	0..1	aggr	This represents the method to provide the content of an element of a diagnostic extended data record.

Table A.88: DiagnosticExtendedDataRecordInterface

Class	DiagnosticExternalAuthenticationInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticExternalAuthenticationInterface			
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.89: DiagnosticExternalAuthenticationInterface

Class	DiagnosticExternalAuthenticationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the external authentication. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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: atpSplittable Tags: atp.Splitkey=process
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: RPortPrototypeInExecutableInstanceRef

Table A.90: DiagnosticExternalAuthenticationPortMapping

Class	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, MultilanguageReferrable, 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.91: DiagnosticFreezeFrame

Class	DiagnosticGenericUdsInterface			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.92: DiagnosticGenericUdsInterface

Class	DiagnosticIndicatorInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.93: DiagnosticIndicatorInterface

Class	DiagnosticIndicatorPortMapping			
Note	Defines to which SWC service ports the DiagnosticIndicator is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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: atp.Splitable 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: atp.UriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.94: DiagnosticIndicatorPortMapping

Class	DiagnosticMapping (abstract)			
Note	Abstract element for different kinds of diagnostic mappings.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
Subclasses	<i>DiagnosticEdrDataProviderMapping</i> , <i>DiagnosticEventToDebounceAlgorithmMapping</i> , <i>DiagnosticEventToEnableConditionGroupMapping</i> , <i>DiagnosticEventToOperationCycleMapping</i> , <i>DiagnosticEventToSecurityEventMapping</i> , <i>DiagnosticEventToTroubleCodeUdsMapping</i> , <i>DiagnosticFimAliasEventGroupMapping</i> , <i>DiagnosticFimAliasEventMapping</i> , <i>DiagnosticInhibitSourceEventMapping</i> , <i>DiagnosticMasterToSlaveEventMapping</i> , <i>DiagnosticProvidedDataMapping</i> , <i>DiagnosticSecureCodingMapping</i> , <i>DiagnosticSovdConfigContentMapping</i> , <i>DiagnosticSwMapping</i> , <i>DiagnosticTroubleCodeUdsToClearConditionGroupMapping</i> , <i>DiagnosticTroubleCodeUdsToTroubleCodeObdMapping</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.95: DiagnosticMapping

Class	DiagnosticMemoryDestination (abstract)			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
Subclasses	<i>DiagnosticMemoryDestinationPrimary</i> , <i>DiagnosticMemoryDestinationUserDefined</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
agingRequiresTestedCycle	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.
clearDtcLimitation	<i>DiagnosticClearDtcLimitationEnum</i>	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.
dtcStatusAvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.
eventDisplacementStrategy	<i>DiagnosticEventDisplacementStrategyEnum</i>	0..1	attr	This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed.
maxNumberOfEventEntries	PositiveInteger	0..1	attr	This attribute fixes the maximum number of event entries in the fault memory.
memoryEntryStorageTrigger	<i>DiagnosticMemoryEntryStorageTriggerEnum</i>	0..1	attr	Describes the trigger to allocate an event memory entry.
statusBitHandlingTestFailedSinceLastClear	<i>DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum</i>	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.
typeOfFreezeFrameRecordNumeration	<i>DiagnosticTypeOfFreezeFrameRecordNumerationEnum</i>	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

Table A.96: DiagnosticMemoryDestination

Class	DiagnosticMemoryDestinationPortMapping			
Note	Defines to which SWC service ports the DiagnosticMemoryDestination. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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.
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.97: DiagnosticMemoryDestinationPortMapping

Class	DiagnosticMemoryDestinationPrimary			
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.98: DiagnosticMemoryDestinationPrimary

Class	DiagnosticMemoryDestinationUserDefined			
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.





Class	DiagnosticMemoryDestinationUserDefined			
authenticationEnabled	DiagnosticAuthRoleProxy	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class. Stereotypes: atpSplittable Tags: atp.Splitkey=authenticationEnabled
memoryId	PositiveInteger	0..1	attr	This represents the identifier of the user-defined memory.

Table A.99: DiagnosticMemoryDestinationUserDefined

Class	DiagnosticMonitorInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.100: DiagnosticMonitorInterface

Class	DiagnosticMonitorPortMapping			
Note	Defines to which SWC service port the Diagnostic Monitor is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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: atpSplittable Tags: atp.Splitkey=process
rPortPrototypeInExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorPortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.101: DiagnosticMonitorPortMapping

Class	DiagnosticMultipleConditionInterface			
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 DiagnosticMultipleConditionInterface allows for handling more than one condition in the scope of a single PortPrototype. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticMultipleConditionInterface			
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.102: DiagnosticMultipleConditionInterface

Class	DiagnosticMultipleConditionPortMapping			
Note	Defines to which SWC service port that can handle a collection of diagnostic conditions the specific condition is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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: atp.Splittable Tags: atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticConditionPortMapping on the AUTOSAR adaptive platform. Stereotypes: atp.UriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.103: DiagnosticMultipleConditionPortMapping

Class	DiagnosticMultipleEventInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.104: DiagnosticMultipleEventInterface

Class	DiagnosticMultipleEventPortMapping			
Note	Defines to which SWC service port that can handle a collection of event status the specific event is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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: atp.Splittable 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: atp.UriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.105: DiagnosticMultipleEventPortMapping

Class	DiagnosticMultipleMonitorInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.106: DiagnosticMultipleMonitorInterface

Class	DiagnosticMultipleMonitorPortMapping			
Note	Defines to which SWC service port that can handle a collection of monitors the specific event is mapped Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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.





Class	DiagnosticMultipleMonitorPortMapping			
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 MonitorMultipleMonitorPortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.107: DiagnosticMultipleMonitorPortMapping

Class	DiagnosticMultipleResourcePortMapping (abstract)			
Note	This abstract base class enables the mapping of diagnostic PortInterfaces that deal with multiple diagnostic resources. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Subclasses	DiagnosticMultipleConditionPortMapping , DiagnosticMultipleEventPortMapping , DiagnosticMultipleMonitorPortMapping			
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.108: DiagnosticMultipleResourcePortMapping

Enumeration	DiagnosticOccurrenceCounterProcessingEnum			
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.109: DiagnosticOccurrenceCounterProcessingEnum

Class	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 , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
type	DiagnosticOperationCycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table A.110: DiagnosticOperationCycle

Class	DiagnosticOperationCycleInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.111: DiagnosticOperationCycleInterface

Class	DiagnosticOperationCyclePortMapping			
Note	Defines to which SWC service ports the DiagnosticOperationCycle 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
operationCycle	DiagnosticOperation Cycle	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: atpSplittable Tags: atp.Splitkey=process This Attribute is only used by the AUTOSAR Adaptive Platform.
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic OperationCyclePortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: RPortPrototypeIn ExecutableInstanceRef This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.112: DiagnosticOperationCyclePortMapping

Class	DiagnosticParameter			
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, 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 AbstractDiagnosticParameter is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let AbstractDiagnosticParameter 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.113: DiagnosticParameter

Class	DiagnosticPeriodicRate			
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.114: DiagnosticPeriodicRate

Class	DiagnosticPortInterface (abstract)			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Subclasses	DiagnosticAbstractDataIdentifierInterface , DiagnosticAbstractRoutineInterface , DiagnosticAuthenticationInterface , DiagnosticComControllInterface , DiagnosticConditionInterface , DiagnosticDTCInformationInterface , DiagnosticDoIPActivationLineInterface , DiagnosticDoIPEntityIdentificationInterface , DiagnosticDoIPGroupIdentificationInterface , DiagnosticDoIPPowerModelInterface , DiagnosticDoIPTTriggerVehicleAnnouncementInterface , DiagnosticDownloadInterface , DiagnosticEcuResetInterface , DiagnosticEventInterface , DiagnosticExtendedDataRecordInterface , DiagnosticExternalAuthenticationInterface , DiagnosticGenericUdsInterface , DiagnosticIndicatorInterface , DiagnosticMonitorInterface , DiagnosticMultipleResourceInterface , DiagnosticOperationCycleInterface , DiagnosticRequestFileTransferInterface , DiagnosticSecurityLevelInterface , DiagnosticServiceValidationInterface , DiagnosticSovdPortInterface , DiagnosticTransmitCertificateInterface , DiagnosticUploadInterface			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
—	—	—	—	—

Table A.115: DiagnosticPortInterface

Class	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=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.





Class	DiagnosticProvidedDataMapping			
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 to which the internal data is mapped, further qualified by the DiagnosticProvidedDataMapping.dataProvider.
dataProvider	NameToken	0..1	attr	This represents the ability to further specify the data provider.

Table A.116: DiagnosticProvidedDataMapping

Class	DiagnosticReadDTCInformation			
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	DiagnosticRead DTCInformationClass	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.117: DiagnosticReadDTCInformation

Class	DiagnosticReadDataByIdentifier			
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.118: DiagnosticReadDataByIdentifier

Class	DiagnosticReadDataByIdentifierClass			
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





Class	DiagnosticReadDataByIdentifierClass			
maxDidToRead	PositiveInteger	0..1	attr	This attribute represents the maximum number of allowed DIDs in a single instance of DiagnosticReadDataByIdentifier for non-OBDDID requests (at least one DID outside the OBD range of 0xF400-0xF8FF).

Table A.119: DiagnosticReadDataByIdentifierClass

Class	DiagnosticReadDataByPeriodicIDClass			
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
maxPeriodicDidToRead	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.
schedulerMaxNumber	PositiveInteger	0..1	attr	This represents the maximum number of periodic data identifiers that can be scheduled in parallel.

Table A.120: DiagnosticReadDataByPeriodicIDClass

Enumeration	DiagnosticRecordTriggerEnum
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.121: DiagnosticRecordTriggerEnum

Class	DiagnosticRequestFileTransferInterface
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 This Class is only used by the AUTOSAR Adaptive Platform.
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable
Aggregated by	ARPackage.element





Class	DiagnosticRequestFileTransferInterface			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.122: DiagnosticRequestFileTransferInterface

Class	DiagnosticResponseOnEvent			
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
responseOnEventAction	DiagnosticResponseOnEventActionEnum	0..1	attr	Defines sub-functions of the service ResponseOnEvent.
responseOnEventClass	DiagnosticResponseOnEventClass	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.123: DiagnosticResponseOnEvent

Enumeration	DiagnosticResponseOnEventActionEnum	
Note	This meta-class has the ability to define sub-functions of the UDS service ResponseOnEvent.	
Aggregated by	DiagnosticResponseOnEvent.responseOnEventAction	
Literal	Description	
clear	Clears the configured events. Tags: atp.EnumerationLiteralIndex=2	
onChangeOfDataIdentifier	Reports based on change of data identifier. Tags: atp.EnumerationLiteralIndex=6	
onComparisonOfValues	Triggered if data condition is met (e.g. RPM over 5000 1/min). Tags: atp.EnumerationLiteralIndex=8	
onDTCStatusChange	Reports based on change of DTC status. Tags: atp.EnumerationLiteralIndex=7	
report	Reports the activated events. Tags: atp.EnumerationLiteralIndex=3	
reportDTCRecordInformationOnDtcStatusChange	Reports the DTC record-related information based on a DTC status change. (Subfunction 0x09) Tags: atp.EnumerationLiteralIndex=5	
reportMostRecentDtcOnStatusChange	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.124: DiagnosticResponseOnEventActionEnum

Class	DiagnosticResponseOnEventClass			
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
maxNumberOfStoredDTCStatusChangedEvents	PositiveInteger	0..1	attr	The maximum number of DTCs that can be stored as DTCs with change status within one ResponseOnEvent SchedulerRate interval.
maxNumChangeOfDataIdentifierEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function onChangeOfDataIdentifier.
maxNumComparisonOfValueEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function onComparisonOfValues.
maxSupportedDIDLength	PositiveInteger	0..1	attr	The maximum number of measurable data bytes allowed for each DID that is used for comparison or data change.
responseOnEventSchedulerRate	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.
storeEventEnabled	Boolean	0..1	attr	Specifies if the storeEvent functionality of the ResponseOnEvent 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.125: DiagnosticResponseOnEventClass

Enumeration	DiagnosticResponseToEcuResetEnum			
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			
respondBeforeReset	Answer to EcuReset service should come before the reset. Tags: atp.EnumerationLiteralIndex=1			

Table A.126: DiagnosticResponseToEcuResetEnum

Class	DiagnosticRoutine			
Note	This meta-class represents the ability to define a diagnostic routine. Tags: atp.recommendedPackage=DiagnosticRoutines			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, 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





Class	DiagnosticRoutine			
requestResult	DiagnosticRequestRoutineResults	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.127: DiagnosticRoutine

Class	DiagnosticRoutineGenericInterface			
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.128: DiagnosticRoutineGenericInterface

Class	DiagnosticRoutineInterface			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.129: DiagnosticRoutineInterface

Class	DiagnosticRoutineSubfunction (abstract)			
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.130: DiagnosticRoutineSubfunction

Class	DiagnosticSecurityAccess			
Note	This represents an instance of the "Security Access" diagnostic service. Tags: atp.recommendedPackage=DiagnosticSecurityAccesss			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
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: atp.Splitable Tags: atp.Splitkey=securityLevel

Table A.131: DiagnosticSecurityAccess

Class	DiagnosticSecurityAccessClass			
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. Tags: atp.recommendedPackage=DiagnosticSecurityAccesss			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
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. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.132: DiagnosticSecurityAccessClass

Class	DiagnosticSecurityLevel			
Note	This meta-class represents the ability to define a security level considered for diagnostic purposes. Tags: atp.recommendedPackage=DiagnosticSecurityLevels			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
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			
numFailedSecurityAccess	PositiveInteger	0..1	attr	This represents the number of failed security accesses after which the delay time is activated.
securityDelayTime	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.133: DiagnosticSecurityLevel

Class	DiagnosticSecurityLevelInterface			
Note	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.134: DiagnosticSecurityLevelInterface

Class	DiagnosticSecurityLevelPortMapping			
Note	Defines to which SWC service ports the DiagnosticSecurityLevel is mapped. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticSecurityLevelMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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.135: DiagnosticSecurityLevelPortMapping

Class	DiagnosticServiceGenericMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticServiceGenericMapping			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
diagnosticServiceInstance	DiagnosticServiceInstance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port.
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticServiceGenericMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: DiagnosticServiceGenericMapping

Class	DiagnosticServiceInstance (abstract)			
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
accessPermission	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 DiagnosticServiceInstance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. Stereotypes: atpAbstract

Table A.137: DiagnosticServiceInstance

Class	DiagnosticServiceValidationConfiguration			
Note	This meta-class has the ability to configure the order of manufacturer/supplier-checks. Tags: atp.recommendedPackage=DiagnosticValueConfigurations This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticServiceValidationConfiguration			
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.
supplier ValidationOrder (ordered)	DiagnosticServiceValidationMapping	*	ref	This reference defines the order in which validations created by supplier are executed.

Table A.138: DiagnosticServiceValidationConfiguration

Class	DiagnosticServiceValidationInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.139: DiagnosticServiceValidationInterface

Class	DiagnosticServiceValidationMapping			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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: atp.UriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atp.Splitable Tags: atp.Splitkey=process

Table A.140: DiagnosticServiceValidationMapping

Class	DiagnosticSession			
Note	This meta-class represents the ability to define a diagnostic session. Tags: atp.recommendedPackage=DiagnosticSessions			
Base	ARElement , ARObject , CollectableElement , DiagnosticCommonElement , Identifiable , MultilanguageReferrable , 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.
p2StarServerMax	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.141: DiagnosticSession

Class	DiagnosticSessionControlClass			
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
s3ServerTimeout	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.142: DiagnosticSessionControlClass

Enumeration	DiagnosticSignificanceEnum
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
occurrence	Issue, which indicates additional information concerning insufficient system behavior. Tags: atp.EnumerationLiteralIndex=1

Table A.143: DiagnosticSignificanceEnum

Class	DiagnosticSovdArrayContentElement
Note	This meta-class represents the ability to define a DiagnosticSovdCompositeContentElement with array semantics. This Class is only used by the AUTOSAR Adaptive Platform.
Base	ARObject , DiagnosticAbstractSovdContent , DiagnosticSovdCompositeContentElement , DiagnosticSovdContentElement , Identifiable , MultilanguageReferrable , Referrable
Aggregated by	DiagnosticSovdArrayContentElement.element , DiagnosticSovdContent.element , DiagnosticSovdRecordContentElement.element





Class	DiagnosticSovdArrayContentElement			
Attribute	Type	Mult.	Kind	Note
element	DiagnosticSovdContentElement	0..1	aggr	This aggregation represents the specification of the nature of an element of an enclosing DiagnosticSovdArrayContentElement. It is only necessary to aggregate one element because the array semantics requires that all elements of the array have the same nature. Tags: xml.sequenceOffset=100
maxNumberOfElements	PositiveInteger	0..1	attr	This attribute specifies the maximum number of elements expected in the array. Tags: xml.sequenceOffset=20
minNumberOfElements	PositiveInteger	0..1	attr	This attribute specifies the minimum number of elements expected in the array. Tags: xml.sequenceOffset=10

Table A.144: DiagnosticSovdArrayContentElement

Class	DiagnosticSovdAuthorizationInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD authorization. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.145: DiagnosticSovdAuthorizationInterface

Class	DiagnosticSovdAuthorizationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD authorization. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement , ARObject , CollectableElement , DiagnosticCommonElement , DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticSovdAuthorizationPortMapping on the AUTOSAR adaptive platform. Stereotypes: atp.UriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atp.Splitable Tags: atp.Splitkey=process

Table A.146: DiagnosticSovdAuthorizationPortMapping

Class	DiagnosticSovdBulkDataInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD bulk data transmission. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.147: DiagnosticSovdBulkDataInterface

Class	DiagnosticSovdBulkDataPortMapping			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdBulkDataInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdBulkData. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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 DiagnosticSovdBulkDataPortMapping on the AUTOSAR adaptive platform. Stereotypes: atp.UriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atp.Splittable Tags: atp.Splitkey=process
serviceInstance	DiagnosticSovdBulk Data	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance.

Table A.148: DiagnosticSovdBulkDataPortMapping

Class	DiagnosticSovdConfigContentMapping			
Note	This mapping associates an SOVD configuration parameter service instance to an SOVD content. Tags: atp.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
content	DiagnosticSovdContent	0..1	ref	This reference identifies the applicable SOVD content.
serviceInstance	DiagnosticSovd ConfigurationParameter	0..1	ref	This reference identifies the applicable service instance.

Table A.149: DiagnosticSovdConfigContentMapping

Class	DiagnosticSovdConfigurationInterface			
Note	This meta-class is used to configure a PortInterface for the exchange of configuration content. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.150: DiagnosticSovdConfigurationInterface

Class	DiagnosticSovdConfigurationPortMapping			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdConfigurationInterface to the corresponding SOVD service instance that is modeled as DiagnosticSovdConfiguration. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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 DiagnosticSovdConfigurationPortMapping on the AUTOSAR adaptive platform. Stereotypes: atp.UriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atp.Splittable Tags: atp.Splitkey=process
serviceInstance	DiagnosticSovd Configuration	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance.

Table A.151: DiagnosticSovdConfigurationPortMapping

Class	DiagnosticSovdContent			
Note	This meta-class represents a data packet exchanged between an SOVD Tester and an SOVD server. The semantics is very close to the semantics of a UDS DID, with some extensions. Tags: atp.recommendedPackage=DiagnosticSovdContents This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticAbstractSovdContent, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
element	DiagnosticSovdContent Element	*	aggr	This aggregation identifies the collection of elements of the enclosing DiagnosticSovdContent. Stereotypes: atp.Splittable Tags: atp.Splitkey=element.shortName

Table A.152: DiagnosticSovdContent

Class	DiagnosticSovdContentElement (abstract)			
Note	This meta-class represents an element of a DiagnosticSovdContent. When compared to the modeling of a UDS DID, this class would correspond to a mixture of DiagnosticParameter and DiagnosticData Element. The concrete semantics of a DiagnosticSovdContentElement is defined by its subclasses. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, DiagnosticAbstractSovdContent, Identifiable , MultilanguageReferrable , Referrable			
Subclasses	DiagnosticSovdCompositeContentElement, DiagnosticSovdPrimitiveContentElement			
Aggregated by	DiagnosticSovdArrayContentElement.element , DiagnosticSovdContent.element , DiagnosticSovdRecord ContentElement.element			
Attribute	Type	Mult.	Kind	Note
isOptional	Boolean	0..1	attr	This attribute defines whether the enclosing Diagnostic SovdContentElement is considered optional. Note that array elements cannot be declared optional.

Table A.153: DiagnosticSovdContentElement

Class	DiagnosticSovdContentPortMapping			
Note	This mapping associates a PPortPrototype typed by either a DiagnosticSovdContentInterface or a DiagnosticSovdContentElementInterface to the corresponding SOVD content that is modeled as DiagnosticAbstractSovdContent. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
content	DiagnosticAbstractSovd Content	0..1	ref	This reference identifies the applicable content for the mapping.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdContentPortMapping 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.

Table A.154: DiagnosticSovdContentPortMapping

Class	DiagnosticSovdData			
Note	This meta-class represents a "Data" SOVD service instance. Tags: atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticSovdService Instance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
sovdData Category	DiagnosticSovdData Category	0..1	ref	This reference identifies the SOVD category that applies to the enclosing SOVD data. Tags: atp.Status=candidate
sovdGroup	DiagnosticSovdGroup	*	ref	This reference identifies the collection of SOVD groups in which the enclosing SOVD data is a "member". Tags: atp.Status=candidate
target	DiagnosticSovdContent	0..1	ref	This reference identifies the content that the Diagnostic SovdData service instance is responsible for.

Table A.155: DiagnosticSovdData

Class	DiagnosticSovdDataCategory			
Note	This meta-class represents the primary filter criterion for the access to SOVD data. Tags: atp.Status=candidate atp.recommendedPackage=DiagnosticDataCategorys This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
group	DiagnosticSovdGroup	*	ref	This reference identifies the SOVD groups in which the SOVD category is a "member". Tags: atp.Status=candidate

Table A.156: DiagnosticSovdDataCategory

Class	DiagnosticSovdFaultMemoryAccess			
Note	This meta-class represents the ability to access (read/get and delete) the fault memory. Tags: atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticSovdService Instance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
—	—	—	—	—

Table A.157: DiagnosticSovdFaultMemoryAccess

Class	DiagnosticSovdGroup			
Note	This meta-class represents the secondary filter criterion for the access to SOVD data Tags: atp.Status=candidate atp.recommendedPackage=DiagnosticSovdGroups This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticSovdGroup			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.158: DiagnosticSovdGroup

Class	DiagnosticSovdLog			
Note	This meta-class represents a "Log" SOVD service instance. Tags: atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticSovdServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.159: DiagnosticSovdLog

Class	DiagnosticSovdMethod			
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.recommendedPackage=DiagnosticSovdMethods This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
delete	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive.
get	DiagnosticSovdMethod Primitive	*	aggr	This represents the "get" method primitive.
post	DiagnosticSovdMethod Primitive	*	aggr	This represents the "post" method primitive.
put	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive.

Table A.160: DiagnosticSovdMethod

Class	DiagnosticSovdOperation			
Note	This meta-class represents a "Operation" SOVD service instance. Tags: atp.recommendedPackage=DiagnosticSovdServiceInstances This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticSovdServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note





Class	DiagnosticSovdOperation			
proximityProofRequired	Boolean	0..1	attr	If true, the execution of the operation requires proof of co-location, and the offline capability description includes the attribute x-sovd-proximity-proof-required in the Path Item object. An SOVD client can use this information to filter operations in advance if a co-location proof cannot be provided. Tags: atp.Status=candidate
readStatus	DiagnosticSovdOperationGetStatus	1	aggr	This aggregation identifies the "read status" operation primitive.
start	DiagnosticSovdOperationStart	0..1	aggr	This aggregation identifies the "start" operation primitive.
terminate	DiagnosticSovdOperationStop	0..1	aggr	This aggregation identifies the "terminate" operation primitive.

Table A.161: DiagnosticSovdOperation

Class	DiagnosticSovdOperationGetStatus			
Note	This class represents the "get status" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARObject</i> , DiagnosticSovdOperationPrimitive			
Aggregated by	DiagnosticSovdOperation.readStatus			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.162: DiagnosticSovdOperationGetStatus

Class	<i>DiagnosticSovdOperationPrimitive</i> (abstract)			
Note	This meta-class represents a primitive of an SOVD operation. The class is modeled as an abstract class to be able to contribute its properties to the concrete subclasses. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARObject</i>			
Subclasses	DiagnosticSovdOperationGetStatus , DiagnosticSovdOperationStart , DiagnosticSovdOperationStop			
Attribute	Type	Mult.	Kind	Note
accessPermission	DiagnosticAccessPermission	0..1	ref	This reference identifies the applicable access permission. Tags: xml.sequenceOffset=100
requestArgument	DiagnosticSovdContent	0..1	ref	This reference identifies the arguments applicable for the operation request. Tags: xml.sequenceOffset=50
responseArgument	DiagnosticSovdContent	0..1	ref	This reference identifies the arguments applicable for the operation response. Tags: xml.sequenceOffset=60

Table A.163: DiagnosticSovdOperationPrimitive

Class	DiagnosticSovdOperationStart			
Note	This class represents the "start" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARObject</i> , DiagnosticSovdOperationPrimitive			
Aggregated by	DiagnosticSovdOperation.start			





Class	DiagnosticSovdOperationStart			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.164: DiagnosticSovdOperationStart

Class	DiagnosticSovdOperationStop			
Note	This class represents the "stop" primitive of an SOVD operation. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, DiagnosticSovdOperationPrimitive			
Aggregated by	DiagnosticSovdOperation.terminate			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.165: DiagnosticSovdOperationStop

Class	DiagnosticSovdPrimitiveContentElement			
Note	This meta-class represents a primitive leaf element of a DiagnosticSovdContent. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, DiagnosticAbstractSovdContent , DiagnosticSovdContentElement , Identifiable , Multilanguage , Referrable , Referrable			
Aggregated by	DiagnosticSovdArrayContentElement.element , DiagnosticSovdContent.element , DiagnosticSovdRecordContentElement.element			
Attribute	Type	Mult.	Kind	Note
swDataDef Props	SwDataDefProps	0..1	aggr	This aggregation represents the applicable data definition properties. Stereotypes: atpSplitable Tags: atp.Splitkey=swDataDefProps xml.sequenceOffset=100
type	DiagnosticSovdPrimitiveContentTypeEnum	0..1	attr	This attribute denotes the type of the primitive content element. Tags: xml.sequenceOffset=10

Table A.166: DiagnosticSovdPrimitiveContentElement

Class	DiagnosticSovdProps			
Note	This meta-class collects common attributes that are specific for SOVD. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject			
Aggregated by	DiagnosticCommonProps.sovdProperties			
Attribute	Type	Mult.	Kind	Note
singleElement Serialization	DiagnosticSovdSerializationEnum	0..1	attr	This attribute defines the serialization strategy that is valid if and only if the contet consists of a single element.

Table A.167: DiagnosticSovdProps

Class	DiagnosticSovdProximityChallengeInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD proximity challenge. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			





Class	DiagnosticSovdProximityChallengeInterface			
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.168: DiagnosticSovdProximityChallengeInterface

Class	DiagnosticSovdProximityChallengePortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD proximity challenge. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
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 SovdProximityChallengePortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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: atpSplittable Tags: atp.Splitkey=process

Table A.169: DiagnosticSovdProximityChallengePortMapping

Class	DiagnosticSovdServiceValidationInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD service validation. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.170: DiagnosticSovdServiceValidationInterface

Class	DiagnosticSovdServiceValidationPortMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD service validation. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			





Class	DiagnosticSovdServiceValidationPortMapping			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdValidationPortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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.

Table A.171: DiagnosticSovdServiceValidationPortMapping

Class	DiagnosticSovdUpdateInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD update procedure. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.172: DiagnosticSovdUpdateInterface

Class	DiagnosticSovdUpdatePortMapping			
Note	This mapping associates a PPortPrototype typed by an DiagnosticSovdUpdateInterface with the corresponding SOVD service instance that is modeled as a DiagnosticSovdUpdate. Tags: atp.recommendedPackage=DiagnosticPortMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdUpdatePortMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
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
serviceInstance	DiagnosticSovdUpdate	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. Tags: atp.Status=candidate

Table A.173: DiagnosticSovdUpdatePortMapping

Enumeration	DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum
Note	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.
Aggregated by	DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear
Literal	Description
statusBitAgingAndDisplacement	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.174: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum

Class	DiagnosticTransmitCertificateInterface			
Note	This meta-class represents the ability to implement the transmit-certificate functionality on application software level. Tags: atp.recommendedPackage=DiagnosticPortInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
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.175: DiagnosticTransmitCertificateInterface

Class	DiagnosticTroubleCodeGroup			
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 , MultilanguageReferrable , 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: atp.Splitable; atp.Variation 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: atp.Variation Tags: vh.latestBindingTime=preCompileTime

Table A.176: DiagnosticTroubleCodeGroup

Class	DiagnosticTroubleCodeProps			
Note	This element defines common Dtc properties that can be reused by different DTCs. Tags: atp.recommendedPackage=DiagnosticTroubleCodePropss			
Base	ARElement , ARObject , CollectableElement , DiagnosticCommonElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note





Class	DiagnosticTroubleCodeProps			
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
diagnosticMemory	DiagnosticMemoryDestination	0..1	ref	Reference to the applicable DiagnosticMemory Destination. Stereotypes: atpSplitable Tags: atp.Splitkey=diagnosticMemory
extendedDataRecord	DiagnosticExtendedDataRecord	*	ref	Defines the links to an extended data class sampler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=extendedDataRecord.diagnosticExtendedDataRecord, extendedDataRecord.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
freezeFrame	DiagnosticFreezeFrame	*	ref	Define the links to a freeze frame class sampler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=freezeFrame.diagnosticFreezeFrame, freezeFrame.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
immediateNvDataStorage	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.
legislatedFreezeFrameContentUdsObd	DiagnosticDataIdentifierSet	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the UDS protocol such as OBD on UDS or WWH-OBD. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=legislatedFreezeFrameContentUdsObd.diagnosticDataIdentifierSet, legislatedFreezeFrameContentUdsObd.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
legislatedSovdFreezeFrameContent	DiagnosticSovdContentGroup	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the SOVD protocol. This Attribute is only used by the AUTOSAR Adaptive Platform.
maxNumberFreezeFrameRecords	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.
obdRelevance	Boolean	0..1	attr	The attribute controls whether the enclosing DTC is relevant for OBD. Stereotypes: atpVariation Tags: 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=postBuild
significance	DiagnosticSignificanceEnum	0..1	attr	Significance of the event, which indicates additional information concerning fault classification and resolution.





Class	DiagnosticTroubleCodeProps			
snapshotRecordContent	DiagnosticDataIdentifierSet	0..1	ref	This represents the freeze frame layout as a set of DIDs. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=snapshotRecordContent.diagnosticDataIdentifierSet, snapshotRecordContent.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
sovdSnapshotContent	DiagnosticSovdContentGroup	0..1	ref	This represents the freeze frame layout as a set of SOVD Content. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.177: DiagnosticTroubleCodeProps

Class	DiagnosticTroubleCodeUds			
Note	This element is used to describe diagnostic trouble codes (DTCs). Tags: atp.recommendedPackage=DiagnosticTroubleCodes			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticTroubleCode, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
considerPtoStatus	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=postBuild
eventReadinessGroup	EventObdReadinessGroup	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.eventObdReadinessGroup, 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=postBuild
obdDtcValue3Byte	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=postBuild
severity	DiagnosticUdsSeverityEnum	0..1	attr	DTC severity according to ISO 14229-1. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
troubleCodeProps	DiagnosticTroubleCodeProps	0..1	ref	Defined properties associated with the DemDTC. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=troubleCodeProps.diagnosticTroubleCodeProps, troubleCodeProps.variationPoint.shortLabel vh.latestBindingTime=postBuild





Class	DiagnosticTroubleCodeUds			
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
wwhObdDtcClass	DiagnosticWwhObdDtcClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OB DTC. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild

Table A.178: DiagnosticTroubleCodeUds

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. Tags: atp.recommendedPackage=DiagnosticMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , Identifiable , MultilanguageReferrable , PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
clearConditionGroup	DiagnosticClearConditionGroup	0..1	ref	This reference identifies the applicable DiagnosticClearConditionGroup.
troubleCodeUds	DiagnosticTroubleCodeUds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping.

Table A.179: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

Enumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum
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.180: DiagnosticTypeOfFreezeFrameRecordNumerationEnum

Class	DiagnosticUploadInterface			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.181: DiagnosticUploadInterface

Class	DiagnosticWriteDataByIdentifier			
Note	This represents an instance of the "Write Data by Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , DiagnosticDataByIdentifier , DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
writeClass	DiagnosticWriteDataByIdentifierClass	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 DiagnosticWriteDataByIdentifier in the given context.

Table A.182: DiagnosticWriteDataByIdentifier

Enumeration	DolpEidRetrievalEnum			
Note	Enumeration with options to retrieve EID. This Enumeration is only used by the AUTOSAR Adaptive Platform.			
Aggregated by	DolpNetworkConfiguration.eidRetrieval , DolpNetworkConfigurationDesign.eidRetrieval			
Literal	Description			
eidUseApi	API DiagnosticDoIPEntityIdentification is used to retrieve eid Tags: atp.EnumerationLiteralIndex=1			
eidUseConfigValue	eid is configured manually by DolpInstantiation.eid Tags: atp.EnumerationLiteralIndex=2			
eidUseMac	MAC of the network interface is used as eid Tags: atp.EnumerationLiteralIndex=0			

Table A.183: DolpEidRetrievalEnum

Class	DolpFunctionalClusterDesign			
Note	This meta-class defines the attributes for the DoIP configuration settings in the MachineDesign. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARObject</i> , <i>AbstractFunctionalClusterDesign</i> , Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	MachineDesign.functionalClusterDesign			
Attribute	Type	Mult.	Kind	Note
dolpLogicalAddress	DolpLogicalAddress	0..1	aggr	This aggregation contains information about the DoIP logical address.
dolpProtocolVersion	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.
entityStatusMaxByteFieldUse	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.
maxRequestBytes	PositiveInteger	0..1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
networkInterface	DolpNetworkConfigurationDesign	*	aggr	This element collects DoIP properties that are network interface specific.





Class	DolpFunctionalClusterDesign			
request Configuration Design	DolpRequest 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.184: DolpFunctionalClusterDesign

Class	DolpInstantiation			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, AdaptiveModuleInstantiation, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , MultilanguageReferrable , NonOsModuleInstantiation, Referrable			
Aggregated by	AtpClassifier.atpFeature, Machine.moduleInstantiation			
Attribute	Type	Mult.	Kind	Note
dolpDesign	DolpFunctionalClusterDesign	0..1	ref	Reference to the DoIP Design that this DolpInstantiation 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	DolpNetworkConfiguration	*	aggr	Network interface specific DoIP properties.
request Configuration	DolpRequest 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.185: DolpInstantiation

Class	DolpNetworkConfiguration			
Note	This element collects DoIP properties that are network interface specific. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject			
Aggregated by	DolpInstantiation.networkInterface			
Attribute	Type	Mult.	Kind	Note
eidRetrieval	DolpEidRetrievalEnum	0..1	attr	This attribute defines how Dolp Entity 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	DolpNetworkConfiguration			
maxInitialVehicleAnnouncementTime	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..maxInitialVehicleAnnouncementTime].
maxTesterConnections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
networkConfiguration	PlatformModuleEthernetEndpointConfiguration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN.
networkConfigurationDesign	DolpNetworkConfigurationDesign	0..1	ref	Reference to the DoIP network configuration design that this DolpNetworkConfiguration implements.
networkInterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheckResponseTimeout	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.
tcpGeneralInactivityTime	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.
tcpInitialInactivityTime	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.
vehicleAnnouncementCount	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2:2019.
vehicleAnnouncementInterval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2019.
vehicleIdentificationSyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement.

Table A.186: DolpNetworkConfiguration

Class	DolpNetworkConfigurationDesign			
Note	This element collects DoIP properties that are network interface specific. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable			
Aggregated by	DolpFunctionalClusterDesign.networkInterface			
Attribute	Type	Mult.	Kind	Note
eidRetrieval	DolpEidRetrievalEnum	0..1	attr	This attribute defines how Dolp Entity Identification is retrieved.
isActivationLineDependent	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	DolpNetworkConfigurationDesign			
maxInitialVehicleAnnouncementTime	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..maxInitialVehicleAnnouncementTime].
maxTesterConnections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
networkConfiguration	PlatformModuleEthernetEndpointConfiguration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN.
networkInterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheckResponseTimeout	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.
tcpGeneralInactivityTime	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.
tcpInitialInactivityTime	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
vehicleAnnouncementCount	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2:2019.
vehicleAnnouncementInterval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2019.
vehicleIdentificationSyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement.

Table A.187: DolpNetworkConfigurationDesign

Class	GenericTpConnection			
Note	A TpConnection that defines a communication between the logical target address of the DoIP Entity and the source address range. Tags: atp.recommendedPackage=GenericTpConnections This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , UploadableDesignElement , UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
priority	PositiveInteger	0..1	attr	This attribute defines the priority for a particular communication connection. For more details see the SWS Diagnostics specification.





Class	GenericTpConnection			
sourceAddressRangeEnd	PositiveInteger	0..1	attr	The DoIP source address defines the logical address of the tester that starts the communication. The GenericTpConnection is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and sourceAddressRangeEnd. This attribute defines the end of the range.
sourceAddressRangeStart	PositiveInteger	0..1	attr	The DoIP source address defines the logical address of the tester that starts the communication. The GenericTpConnection is able to define a source address range (without gaps) with the two attributes sourceAddressRangeStart and sourceAddressRangeEnd. This attribute defines the start of the range.

Table A.188: GenericTpConnection

Class	Identifiable (abstract)
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
Subclasses	ARPackage, AbstractDolpLogicAddressProps, AbstractEvent, AbstractFunctionalClusterDesign, AbstractImplementationDataTypeElement, AbstractSecurityEventFilter, AbstractSecurityIdsmInstanceFilter, AbstractServiceInstance, AbstractSignalBasedToSignalTriggeringMapping, AdaptiveSwcInternalBehavior, ApApplicationEndpoint, ApmcAbstractDefinition, ApmcConfigurationElementDef, ApmcContainerElementValue, ApmcContainerValue, ApmcEnumerationLiteralDef, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpFeature, AutosarOperationArgumentInstance, AutosarVariableInstance, BuildActionEntity, BuildActionEnvironment, Chapter, CheckpointTransition, ClientIdDefinition, ClientServerOperation, Code, CollectableElement, ComManagementMapping, CommConnectorPort, CommunicationConnector, CommunicationController, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, CouplingPortAbstractShaper, CouplingPortStructuralElement, CryptoCertificate, CryptoCertificateGroup, CryptoKeySlot, CryptoKeySlotDesign, CryptoKeySlotUsageDesign, CryptoProvider, CryptoServiceMapping, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, DataTransformation, DdsAbstractServiceInstanceElementCp, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, DiagEventDebounceAlgorithm, DiagnosticAbstractSovdContent, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticExtendedDataRecordElement, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, DiagnosticRoutineSubfunction, DiagnosticSovdMethodPrimitive, DltApplication, DltArgument, DltArgumentProps, DltMessage, DolpInterface, DolpLogicAddress, DolpLogicalAddress, DolpNetworkConfigurationDesign, DolpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EthernetWakeupSleepOnDataLineConfig, EventHandler, EventMapping, ExclusiveArea, ExecutableEntity, ExecutionTime, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMethodMapping, FlexrayArTpNode, FlexrayTpPduPool, FrameTriggering, GeneralParameter, GlobalSupervision, GlobalTimeGateway, GlobalTimeMaster, GlobalTimeSlave, HealthChannel, HeapUsage, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IEEE1722TpAcfBus, IEEE1722TpAcfBusPart, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, IdentCaption, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, MethodMapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, ModeSwitchSenderComSpecProps, NetworkEndpoint, NmCluster, NmNode, PackageableElement, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, PersistenceDeploymentElement, PersistenceInterfaceElement, PhmSupervision, PhysicalChannel, PortGroup, PortInterfaceMapping, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, QueuedReceiverComSpecProps, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, SdgAttribute, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, SecureCommunicationDeployment, Secure





Class	Identifiable (abstract)			
	△ CommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, ServerComSpecProps, ServiceInterfaceDeploymentElement, ServiceInterfaceElementSecureComConfig, ServiceNeeds, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, StackUsage, StateManagementActionItem, StateManagementActionList, StateManagementStateNotification, StateManagementStateRequest, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionModeCondition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, TimeBaseResource, TimingClock, TimingClockSyncAccuracy, TimingCondition, TimingConstraint, TimingDescription, TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, TracedFailure, TransformationISignalPropsIdent, TransformationProps, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, UriDescription, 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: atpSplittable Tags: atp.Splitkey=adminData xml.sequenceOffset=-40
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	MultiLanguageOverviewParagraph	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 ▽





Class	<i>Identifiable</i> (abstract)			
				△ 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.189: Identifiable

Class	<i>MultilanguageReferrable</i> (abstract)			
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.190: MultilanguageReferrable

Class	PlatformModuleEthernetEndpointConfiguration			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, Identifiable , MultilanguageReferrable , PackageableElement , 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	Definition 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.191: PlatformModuleEthernetEndpointConfiguration

Class	<i>PortInterface</i> (abstract)			
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			





Class	PortInterface (abstract)			
Subclasses	<i>AbstractRawDataStreamInterface</i> , <i>AbstractSuspendToRamInterface</i> , <i>AbstractSynchronizedTimeBaseInterface</i> , <i>ClientServerInterface</i> , <i>CryptoInterface</i> , <i>DataInterface</i> , <i>DiagnosticPortInterface</i> , <i>FirewallStateSwitchInterface</i> , <i>IdsmAbstractPortInterface</i> , <i>LogAndTraceInterface</i> , <i>ModeSwitchInterface</i> , <i>NetworkManagementPortInterface</i> , <i>PersistencyInterface</i> , <i>PlatformHealthManagementInterface</i> , <i>ServiceInterface</i> , <i>StateManagementPortInterface</i> , <i>TriggerInterface</i>			
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 This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.192: PortInterface

Class	PortPrototype (abstract)			
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.			
Base	<i>ARObject</i> , <i>AtpBlueprintable</i> , <i>AtpFeature</i> , <i>AtpPrototype</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
Subclasses	<i>AbstractProvidedPortPrototype</i> , <i>AbstractRequiredPortPrototype</i>			
Aggregated by	<i>AtpClassifier.atpFeature</i> , <i>SwComponentType.port</i>			
Attribute	Type	Mult.	Kind	Note
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServer Annotation	IoHwAbstractionServerAnnotation	*	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 volatile data port.
parameterPort Annotation	ParameterPortAnnotation	*	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. This Attribute is only used by the AUTOSAR Adaptive Platform.
senderReceiver Annotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication. Stereotypes: atpSplitable Tags: atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.193: PortPrototype

Class	RPortPrototype			
Note	Component port requiring a certain port interface.			
Base	<i>ARObject</i> , <i>AbstractRequiredPortPrototype</i> , <i>AtpBlueprintable</i> , <i>AtpFeature</i> , <i>AtpPrototype</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PortPrototype</i> , <i>Referrable</i>			
Aggregated by	<i>AtpClassifier.atpFeature</i> , <i>SwComponentType.port</i>			
Attribute	Type	Mult.	Kind	Note
required Interface	PortInterface	0..1	tref	The interface that this port requires. Stereotypes: isOfType

Table A.194: RPortPrototype

Class	Referrable (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	<i>ARObject</i>			
Subclasses	<i>AtpDefinition</i> , <i>BswDistinguishedPartition</i> , <i>BswModuleCallPoint</i> , <i>BswModuleClientServerEntry</i> , <i>BswVariableAccess</i> , <i>CouplingPortTrafficClassAssignment</i> , <i>CpplImplementationDataTypeContextTarget</i> , <i>DiagnosticEnvModeElement</i> , <i>EthernetPriorityRegeneration</i> , <i>ExclusiveAreaNestingOrder</i> , <i>HwDescriptionEntity</i> , <i>ImplementationProps</i> , <i>ModeTransition</i> , <i>MultilanguageReferrable</i> , <i>NmNetworkHandle</i> , <i>PncMappingIdent</i> , <i>SingleLanguageReferrable</i> , <i>SoConIPdulIdentifier</i> , <i>SomeipRequiredEventGroup</i> , <i>TpConnectionIdent</i>			
Attribute	Type	Mult.	Kind	Note
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.195: Referrable

Class	SoftwareCluster			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i> , <i>UploadableDeploymentElement</i> , <i>UploadablePackageElement</i>			
Aggregated by	<i>ARPackage.element</i>			
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
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





Class	SoftwareCluster			
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackageElement and that contribute to the completeness of the definition of the SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=containedARElement
contained Package Element	UploadablePackageElement	*	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.
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all SoftwareClusterDesigns applicable for the enclosing SoftwareCluster. Stereotypes: atpUriDef
diagnostic Deployment Props	SoftwareClusterDiagnosticDeploymentProps	0..1	ref	This reference identifies the applicable SoftwareClusterDiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster.
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	AdaptiveModuleInstantiation	*	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	CryptoServiceCertificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature.
version	StrongRevisionLabelString	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table A.196: SoftwareCluster

Class	SoftwareClusterDiagnosticAddress (abstract)			
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. ...). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject			
Subclasses	SoftwareClusterSovdAddress , SoftwareClusterUdsDiagnosticAddress			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	Type	Mult.	Kind	Note





Class	SoftwareClusterDiagnosticAddress (abstract)			
address Semantics	SoftwareClusterDiagnosticAddressSemanticsEnum	0..1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table A.197: SoftwareClusterDiagnosticAddress

Enumeration	SoftwareClusterDiagnosticAddressSemanticsEnum
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster. This Enumeration is only used by the AUTOSAR Adaptive Platform.
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.198: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	SoftwareClusterDiagnosticDeploymentProps			
Note	This meta-class acts as the owner of all deployment-related diagnostic properties of a SoftwareCluster. Tags: atp.recommendedPackage=SoftwareClusterDiagnosticProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARElement, ARObject, CollectableElement, Identifiable , MultilanguageReferrable , PackageableElement , Referrable , UploadableDeploymentElement , UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
diagnostic Extract	DiagnosticContributionSet	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	DiagnosticServiceValidationConfiguration	0..1	aggr	This aggregation represents the ability to define the order of manufacturer and supplier validations in diagnostic management.

Table A.199: SoftwareClusterDiagnosticDeploymentProps

Class	SoftwareClusterSovdAddress			
Note	This meta-class represents the ability to define a diagnostic address specifically for the SOVD case. Tags: atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
Base	ARObject, SoftwareClusterDiagnosticAddress			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	Type	Mult.	Kind	Note





Class	SoftwareClusterSovdAddress			
component Qualifier	String	0..1	attr	This attribute is used to specify the component qualifier for the usage in an SOVD query. Stereotypes: atpIdentityContributor Tags: atp.Status=candidate

Table A.200: SoftwareClusterSovdAddress

Class	SovdModuleInstantiation (abstract)			
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 This Class is only used by the AUTOSAR Adaptive Platform.			
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.201: SovdModuleInstantiation

Class	SovdServerInstantiation			
Note	This meta-class represents the configuration of an SOVD server. Tags: atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.			
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.202: SovdServerInstantiation

Class	«atpVariation» SwDataDefProps			
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, swPointerTargetProps, 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, DiagnosticExtendedDataRecordElement.swDataDefProps, DiagnosticSovdPrimitiveContentElement.swDataDefProps, DltArgumentProps.networkRepresentation, FlatInstanceDescriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance. resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, Receiver ComSpec.networkRepresentation, SecurityEventContextDataElement.networkRepresentation, Sender ComSpec.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDef Props, SystemSignal.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
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





Class	«atpVariation» SwDataDefProps			
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
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





Class	«atpVariation» SwDataDefProps			
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
swIntendedResolution	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
swInterpolationMethod	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
swPointerTargetProps	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. Tags: xml.sequenceOffset=280
swRecordLayout	SwRecordLayout	0..1	ref	Record layout for this data object. Tags: xml.sequenceOffset=290
swRefreshTiming	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
swValueBlockSize	Numerical	0..1	attr	This represents the size of a Value Block Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=80





Class	«atpVariation» SwDataDefProps			
swValueBlockSizeMult (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 swValueBlockSize 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
valueAxisDataType	ApplicationPrimitiveDataType	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.203: SwDataDefProps

Class	SymbolProps			
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, TraceSwitchConfig.namespace			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.204: SymbolProps

Class	ValueGroup			
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	MultilanguageLongName	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.205: 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:	<code>#include "apext/diag/uds_transport/protocol_handler.h"</code>
Forwarding header file:	<code>#include "apext/diag/diag_fwd.h"</code>
Scope:	namespace <code>apext::diag::uds_transport</code>
Symbol:	<code>UdsTransportProtocolHandler</code>
Syntax:	<code>class UdsTransportProtocolHandler { ... };</code>
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:	<code>#include "apext/diag/uds_transport/protocol_handler.h"</code>				
Forwarding header file:	<code>#include "apext/diag/diag_fwd.h"</code>				
Scope:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>				
Symbol:	<code>InitializationResult</code>				
Underlying type:	<code>std::uint8_t</code>				
Syntax:	<code>enum class InitializationResult : std::uint8_t { ... };</code>				
Values:	<table> <tr> <td><code>kInitializeOk</code></td><td><code>= 0</code></td></tr> <tr> <td></td><td>Initialization ok.</td></tr> </table>	<code>kInitializeOk</code>	<code>= 0</code>		Initialization ok.
<code>kInitializeOk</code>	<code>= 0</code>				
	Initialization ok.				





	kInitializeFailed	= 1
		Initialization failed.
Description:	Result of <code>Initialize</code> 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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>
Symbol:	<code>transportprotocolManager</code>
Type:	<code>UdsTransportProtocolMgr &</code>
Syntax:	<code>UdsTransportProtocolMgr& transportprotocolManager;</code>
Description:	The <code>UdsTransportProtocolMgr</code> used/provided by the DM.
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>
Syntax:	<code>virtual ~UdsTransportProtocolHandler () noexcept=default;</code>
Exception Safety:	exception safe





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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>	
Syntax:	<code>explicit UdsTransportProtocolHandler (const UdsTransportProtocolHandlerID handlerId, UdsTransportProtocolMgr &transportProtocolMgr) noexcept;</code>	
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 implementation(s).
	transportProtocolMgr	Reference to UdsTransportProtocolMgr owned by this DM, with which UdsTransportProtocolHandler instance shall interact.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>	





Syntax:	virtual UdsTransportProtocolHandlerID GetHandlerID () const noexcept;	
Return value:	UdsTransportProtocolHandlerID	UdsTransportProtocolHandlerID.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
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 <code>ara::core::Result< UdsTransportProtocolPeriodicHandler & ></code> GetPeriodicHandler () noexcept;	
Return value:	<code>ara::core::Result< UdsTransportProtocolPeriodicHandler & ></code>	If periodic transmissions are supported on this transport protocol, a reference to <code>UdsTransportProtocolPeriodicHandler</code> is returned, and if not supported, an error (kConfigurationMismatch).
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	kConfigurationMismatch	
Description:	Returns the corresponding periodic Transport Protocol 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	<code>kInitializeOk</code> , in case, the initialization was successful, else <code>kInitializeFailed</code> is returned.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Initializes UDS Transport Protocol handler. This methods needs to be called before a call of <code>Start()</code> . It is meant to separate the actual initialization of the UDS 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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>	
Syntax:	<code>virtual bool NotifyReestablishment (ChannelID channelId) noexcept=0;</code>	
Parameters (in):	channelId	The channelId, whose re-establishment shall be notified to UdsTransportProtocolMgr
Return value:	bool	TRUE, if notification request is accepted and can be fulfilled, else FALSE
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Tells the UdsTransportProtocolHandler, to notify the DM core via <code>UdsTransportProtocolMgr::ChannelReestablished()</code> if the given channel has been re-established after next <code>UdsTransportProtocolHandler::Start()</code>. 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 (GlobalChannel Identifier), then the ECU-Reset takes place and after ECU-Restart, all UdsProtocol Handlers/-plugins get restarted via call to <code>UdsTransportProtocolHandler::Start()</code>. 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 <code>UdsTransportProtocolMgr::ChannelReestablished(GlobalChannelIdentifier)</code>. 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 <code>ChannelID</code>. 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 <code>ChannelID</code> 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 Service Identifier request is received on channel identified by the DM calls <code>UdsTransportProtocolHandler::NotifyReestablishment()</code> on this <code>ChannelID</code>. Now the implementation of UdsProtocolHandler stores non-volatile in the context of this call: • NetworkEndpoint (IP-address and port number) of the channel. • NetworkEndpoint (IP-address and port number) of the local port (because in this example, the UdsTransportProtocolHandler listens on/supports different ports). • <code>ChannelID</code> it has currently assigned. After restart this <code>ChannelID</code> is only reused for a channel with exactly the same Network Endpoint addresses as stored non-volatile. If this <code>ChannelID</code> then gets reassigned, then <code>UdsTransportProtocolMgr::ChannelReestablished()</code> has to be called.
Notes:	The UdsTransportProtocolHandler will call <code>UdsTransportProtocolMgr::ChannelReestablished()</code> after <code>UdsTransportProtocolHandler::Start()</code> is called and as soon as it detects that the remote client/tester is reachable again.

]

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:	<code>class apext::diag::uds_transport::UdsTransportProtocolHandler</code>
Syntax:	<code>virtual void Start () noexcept=0;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
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 apex::diag::uds_transport::UdsTransportProtocolHandler::StopUpstream requirements: [RS_Diag_04147](#), [RS_Diag_04168](#)

[

Kind:	function
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"
Scope:	<code>class apex::diag::uds_transport::UdsTransportProtocolHandler</code>
Syntax:	<code>virtual void Stop () noexcept=0;</code>
Return value:	None
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Method to indicate that this UdsTransportProtocolHandler should terminate. If UdsTransportProtocolHandler has stopped, it shall call <code>UdsTransportProtocolMgr::HandlerStopped(UdsTransportProtocolHandlerID)</code>. After return from Stop(), the handler-plugin shall NOT call to UdsTransportProtocolMgr with any other method but <code>UdsTransportProtocolMgr::HandlerStopped()</code>.

]

C.1.1.3.3.7 Transmit

[SWS_DM_00327] Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::TransmitUpstream requirements: [RS_Diag_04147](#), [RS_Diag_04168](#)

[

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_handler.h"	
Scope:	<code>class apex::diag::uds_transport::UdsTransportProtocolHandler</code>	
Syntax:	<code>virtual void Transmit (UdsMessageConstPtr message, ChannelID channel Id) noexcept=0;</code>	
Parameters (in):	message	The message to be transmitted as a UdsMessage::Ptr (unique_ptr style). UdsTransportProtocolHandler has to give back this Uds Message::Ptr via <code>UdsTransportProtocolMgr::Transmit Confirmation()</code> 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:	not thread-safe	
Description:	Transmit a UDS message via the underlying UDS Transport Protocol channel.	
Notes:	This transmit API covers T_Data.req of ISO 14229-2[25] 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:	<code>class UdsTransportProtocolMgr {...};</code>
Description:	The UdsTransportProtocolMgr class manages the interaction between the Diagnostic Server instance(s) and various UDS Transport Protocols (e.g., DoIP[2], CAN[21]). It provides a standard, extensible API for handling transport-specific message reception and optional early payload access.

]

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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>
Symbol:	GlobalChannelIdentifier
Syntax:	<code>using GlobalChannelIdentifier = std::tuple<UdsTransportProtocolHandlerID, ChannelID>;</code>
Description:	Type of tuple to pack UdsTransportProtocolHandlerID and ChannelID together, to form a global unique (among all used UdsTransportProtocolHandlers within DM) identifier of a UdsTransportProtocol channel.

]

C.2.1.1.2 Enumeration: IndicationResult**[SWS_DM_00384] Definition of API enum apex::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 apex::diag::uds_transport::UdsTransportProtocolMgr	
Symbol:	IndicationResult	
Underlying type:	std::uint8_t	
Syntax:	enum class IndicationResult : std::uint8_t {...};	
Values:	kIndicationOk	= 0 Indication Ok.
	kIndicationOccupied	= 1 Indication Occupied.
	kIndicationOverflow	= 2 Indication Overflow.
	kIndicationUnknownTargetAddress	= 3 Indication UnknownTargetAddress.
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 apex::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 apex::diag::uds_transport::UdsTransportProtocolMgr	
Symbol:	TransmissionResult	
Underlying type:	std::uint8_t	
Syntax:	enum class TransmissionResult : std::uint8_t {...};	
Values:	kTransmitOk	= 0 Transmission Ok.
	kTransmitFailed	= 1 Transmission Failed.





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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>
Syntax:	<code>virtual ~UdsTransportProtocolMgr () noexcept=default;</code>
Exception Safety:	exception safe
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>		
Syntax:	<code>virtual void ChannelReestablished (GlobalChannelIdentifier globalChannelId) noexcept=0;</code>		
Parameters (in):	<table border="1"><tr><td><code>globalChannelId</code></td><td>transport protocol channel, which is available again.</td></tr></table>	<code>globalChannelId</code>	transport protocol channel, which is available again.
<code>globalChannelId</code>	transport protocol channel, which is available again.		
Return value:	None		
Exception Safety:	exception safe		





Thread Safety:	not thread-safe
Description:	Notification call from the given transport protocol 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 <code>UdsTransportProtocolHandler::NotifyReestablishment()</code> has to be done. See further documentation at <code>UdsTransportProtocolHandler::NotifyReestablishment()</code> .

]

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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>	
Syntax:	<code>virtual void HandleMessage (UdsMessagePtr message) noexcept=0;</code>	
Parameters (in):	message	The UDS message ptr (unique_ptr semantics) with the request. Ownership of the UDS Message is given back to the generic DM core here.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Hands over a valid received UDS message (currently this is only a request type) from the Transport Layer to the Session Layer. This corresponds to T_Data.ind of Figure 2 from ISO 14229-2[25]. The behavior is asynchronously, i.e. the UDS Message is handed over to the Session Layer and it is expected, that it "instantly" returns, which means, that real processing of the message shall be done asynchronously by the Transport Layer!	

]

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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>	





Syntax:	virtual void HandlerStopped (UdsTransportProtocolHandlerID handlerId) noexcept=0;	
Parameters (in):	handlerId	indication, which plugin stopped.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Notification call 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 <code>UdsTransportProtocolHandler::Stop()</code> .	

]

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 the message.
	targetAddr	UDS target address of the message.
	type	Indication whether the request is <code>kPhysical</code> or <code>kFunctional</code> addressed.
	globalChannelId	Transport Protocol Channel on which the message start happened.
	size	Size in bytes of the UdsMessage starting from <code>SID</code> .
	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 Transport Protocol provides at least the first two bytes of the request message, so the DM can identify a functional Tester Present. Lifetime: Valid until Promise is fulfilled or the processing is cancelled by the cancellationHandler.
Return value:	std::pair< IndicationResult, UdsMessagePtr >	Pair of <code>IndicationResult</code> and a pointer to UDS Message owned/created by DM core and returned to the handler to get filled.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Indicates a message start. This is an interface, which is just served/called by UdsTransportProtocolHandlers, which return <code>TRUE</code> from <code>UdsTransportProtocolHandlers::isStartOfMessageIndicationSupported()</code> .	

]

C.2.1.2.2.5 NotifyMessageFailure**[SWS_DM_00310] Definition of API function apex::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 apex::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void NotifyMessageFailure (UdsMessagePtr message) noexcept=0;	
Parameters (in):	message	Pointer to the UDS Message handed back over to the Session Layer.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
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 apex::diag::uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation**Upstream requirements: [RS_Diag_04215](#)

[

Kind:	function	
Header file:	#include "apext/diag/uds_transport/protocol_mgr.h"	
Scope:	class apex::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:	not thread-safe	
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolMgr</code>	
Syntax:	virtual void TransmitConfirmation (UdsMessageConstPtr message, TransmissionResult result) noexcept=0;	
Parameters (in):	message	For which message (created in <code>IndicateMessage()</code>) this is the confirmation.
	result	Result of transmission. In case the UDS message could be transmitted on network layer <code>kTransmitOk</code> , else <code>kTransmitFailed</code> .
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Notification about the outcome of a transmit request called by core DM at the handler via <code>UdsTransportProtocolHandler::Transmit()</code> . This transmit API covers T_Data.con of ISO 14229-2[25] 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:	<code>class UdsTransportProtocolPeriodicHandler {...};</code>
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler</code>
Syntax:	<code>virtual ~UdsTransportProtocolPeriodicHandler () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler</code>	
Syntax:	<code>virtual std::size_t GetMaxPayloadLength (ChannelID channelId) const noexcept=0;</code>	
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:	not thread-safe	
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:	<code>class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler</code>	
Syntax:	<code>virtual std::size_t GetNumberOfPeriodicMessages (ChannelID channelId) const noexcept=0;</code>	
Parameters (in):	channelId	The concrete connection to send over.
Return value:	std::size_t	Number of periodic messages.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Reports the Transport Protocol 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:	<code>class apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler</code>	
Syntax:	<code>virtual void PeriodicTransmit (ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages) noexcept=0;</code>	
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:	not thread-safe	
Description:	Sends all the messages in the list in the given order. If one message transmission fails, the send process is stopped and the PeriodicTransmit Confirmation is invoked with the number of sent messages and the same Vector with Uds Messages.	

]

C.4 Header: apex/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 apex::diag::uds_transport::ByteSpan

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	type alias
Header file:	#include "apex/diag/uds_transport/protocol_types.h"
Scope:	namespace apex::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 apex::diag::uds_transport::ChannelID

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	type alias
Header file:	#include "apex/diag/uds_transport/protocol_types.h"
Scope:	namespace apex::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 apex::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 apex::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 apex::diag::uds_transport::UdsMessageConstPtr

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	namespace apex::diag::uds_transport
Symbol:	UdsMessageConstPtr
Syntax:	using UdsMessageConstPtr = std::unique_ptr<const UdsMessage, std::function<void(const UdsMessage*)>>;
Thread Safety:	not thread-safe
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 apex::diag::uds_transport::UdsMessagePtr

Upstream requirements: [RS_Diag_04147](#)

[

Kind:	type alias
Header file:	#include "apext/diag/uds_transport/uds_message.h"
Scope:	namespace apex::diag::uds_transport
Symbol:	UdsMessagePtr
Syntax:	using UdsMessagePtr = std::unique_ptr<UdsMessage, std::function<void(UdsMessage*)>>>;
Thread Safety:	not thread-safe
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 apex::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 apex::diag::uds_transport
Symbol:	UdsMessage
Syntax:	class UdsMessage {...};
Description:	class represents an UDS message exchanged between DM generic core (UdsTransportProtocolMgr) 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, the UdsTransportProtocolHandler can 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:	<code>class apext::diag::uds_transport::UdsMessage</code>
Symbol:	Address
Syntax:	<code>using Address = std::uint16_t;</code>
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:	<code>class apext::diag::uds_transport::UdsMessage</code>
Symbol:	MetaInfoMap
Syntax:	<code>using MetaInfoMap = ara::core::Map<ara::core::String, ara::core::String>;</code>
Description:	Type for the meta information attached to a UdsMessage.

]

C.5.2.1.3 Enumeration: TargetAddressType**[SWS_DM_00296] Definition of API enum apex::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 apex::diag::uds_transport::UdsMessage	
Symbol:	TargetAddressType	
Underlying type:	std::uint8_t	
Syntax:	enum class TargetAddressType : std::uint8_t {...};	
Values:	kPhysical	= 0
		UdsMessage is physical addressed.
	kFunctional	= 1
		UdsMessage is functional addressed.
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 apex::diag::uds_transport::UdsMessage::~~UdsMessage***Upstream requirements:* [RS_Diag_04147](#)

[

Kind:	function	
Header file:	#include "apext/diag/uds_transport/uds_message.h"	
Scope:	class apex::diag::uds_transport::UdsMessage	
Syntax:	virtual ~UdsMessage () noexcept=default;	
Exception Safety:	exception safe	
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:	<code>class apext::diag::uds_transport::UdsMessage</code>	
Syntax:	<code>virtual void AddMetaInfo (MetaInfoMap metaInfo) noexcept;</code>	
Parameters (in):	metaInfo	MetaInfo of the request.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	unsafe	
Description:	add new metaInfo to this message.	
Notes:	typically called by the Transport Protocol plugin to add channel specific meta-info. See further MetaInfo related 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:	<code>class apext::diag::uds_transport::UdsMessage</code>	
Syntax:	<code>virtual uds_transport::ByteSpan & GetPayload () noexcept;</code>	
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 implementation to fill the UdsMessage with data in RX path. I.e. UdsTransportProtocolHandler implementation gets the UdsMessage instance from a call to <code>UdsTransportProtocolMgr::IndicateMessage()</code> and then calls this method on it and writes into the returned <code>uds_transport::ByteVector</code> . Marked as "unsafe" with regard to thread safety as implementation is allowed to do resource 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::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 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 thread-safety, as the implementation is allowed to do a resource 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:	<code>class apext::diag::uds_transport::UdsMessage</code>
Syntax:	<code>UdsMessage () noexcept;</code>
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	non public default constructor. The default constructor is protected as we want to forbid, that UdsTransportProtocolHandlers 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

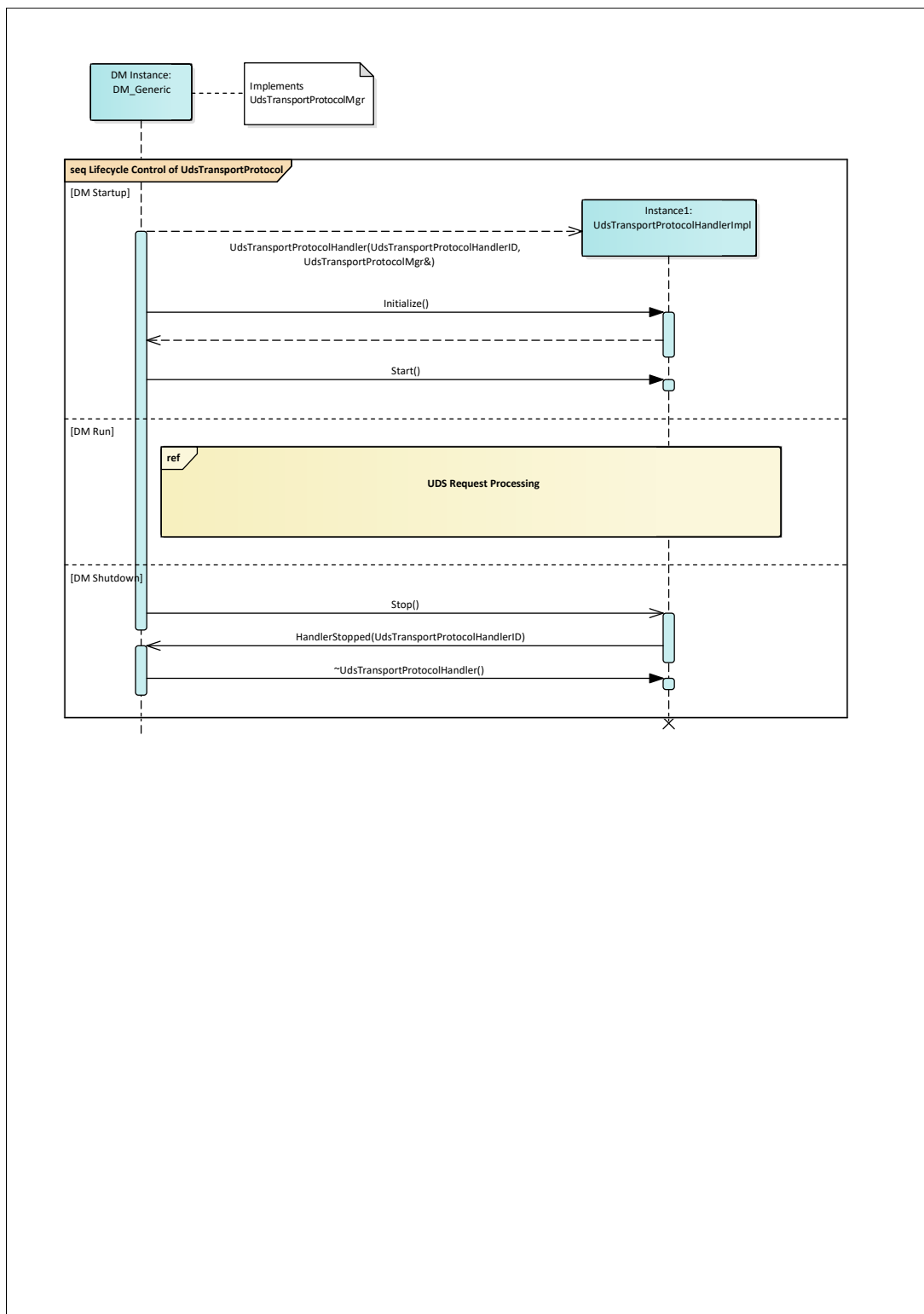


Figure C.1: UDS Transport Lifecycle

C.6.2 UDS Request Processing

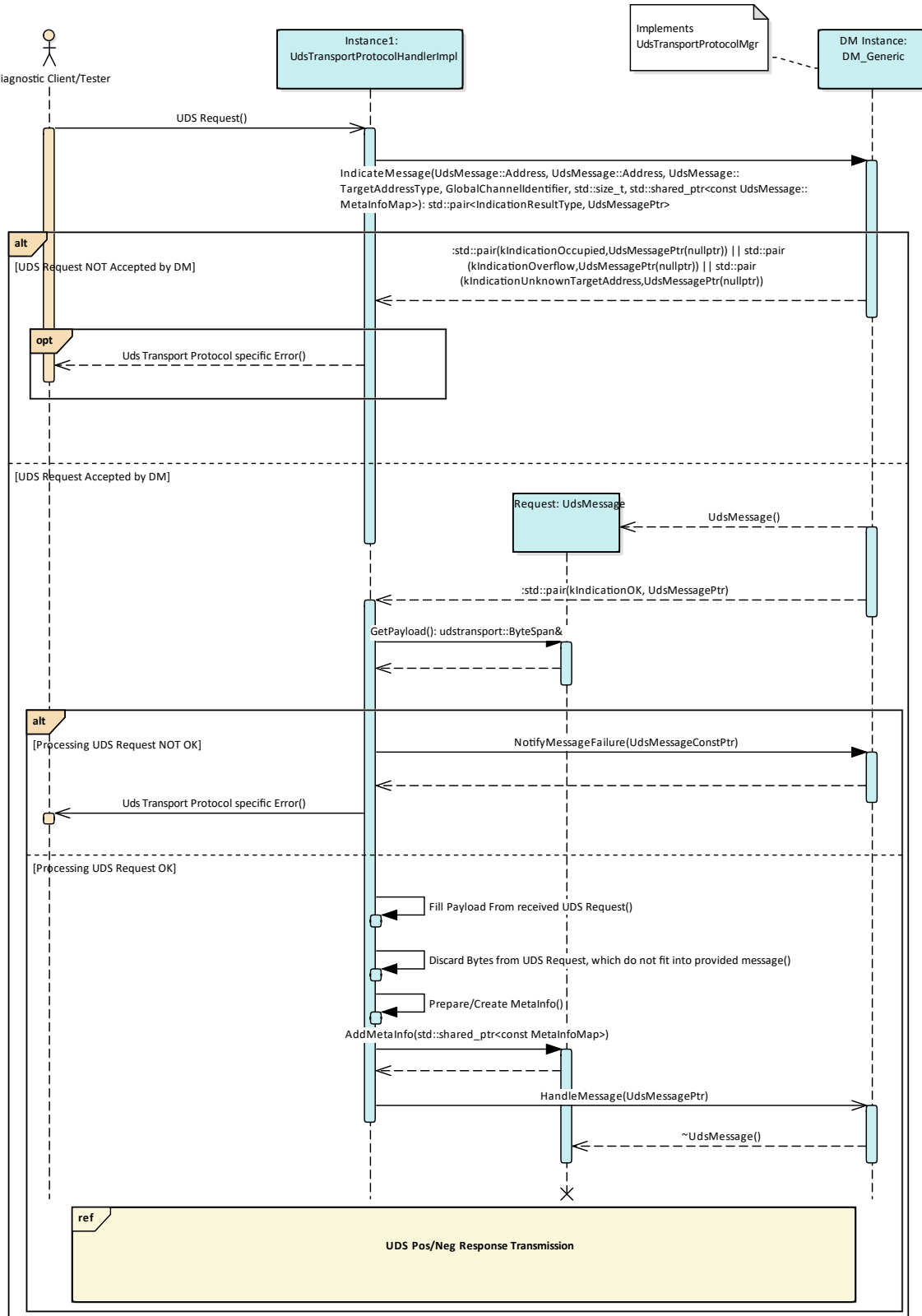


Figure C.2: UDS Transport Request Processing

C.6.3 UDS Response Transmission

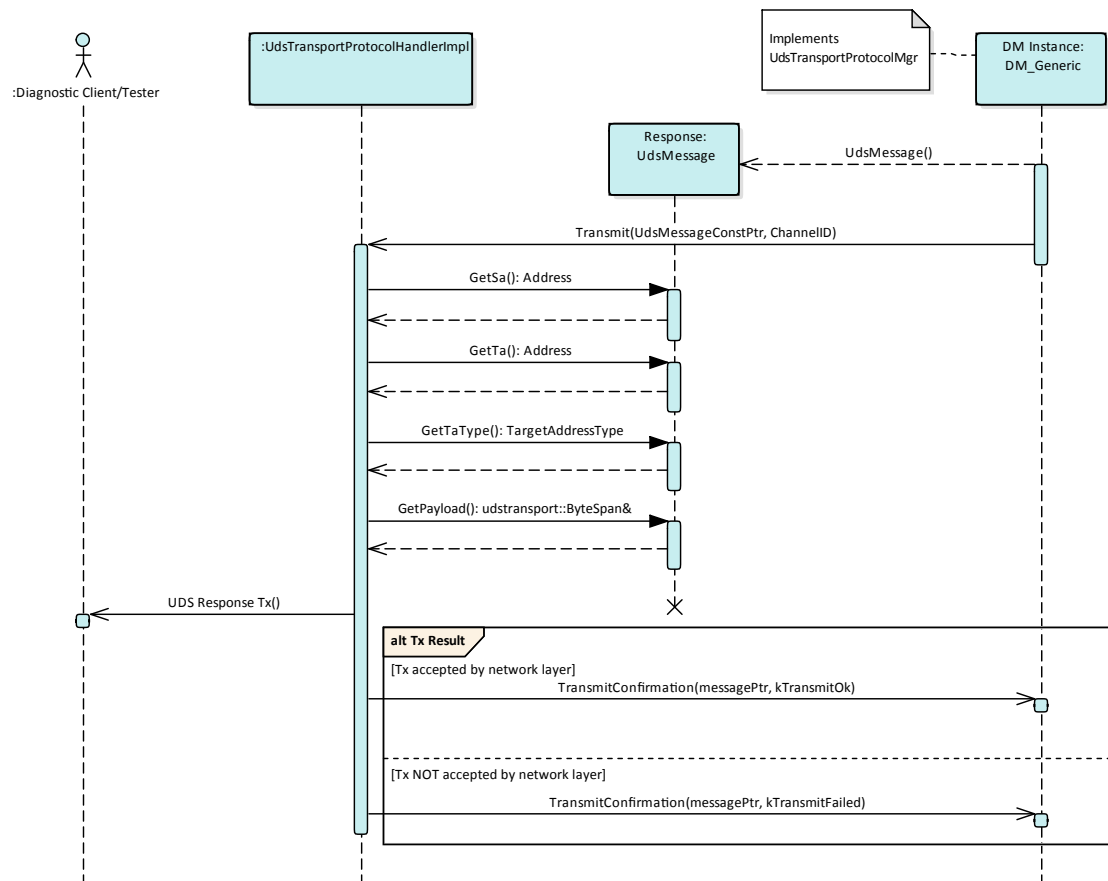


Figure C.3: UDS Response Transmission

C.6.4 Channel Reestablishment

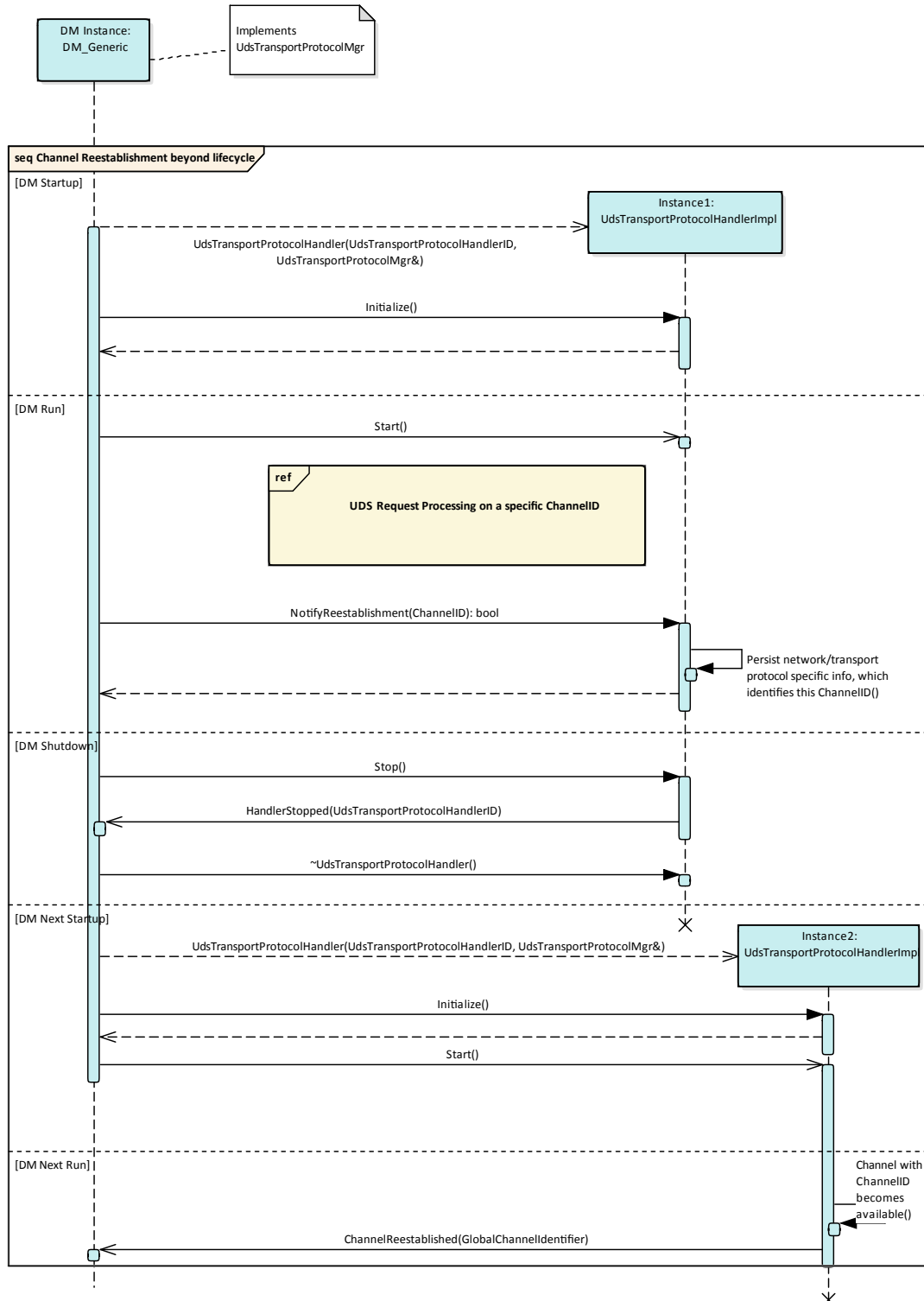


Figure C.4: UDS Transport Channel Reestablishment

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_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





Number	Heading
[SWS_DM_00343]	Acceptance of UDS message reception
[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
[SWS_DM_00073]	Checking enable conditions after status reports





Number	Heading
[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) resources
[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_00397]	Retrieving data for external DiagnosticDataElements
[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
[SWS_DM_00413]	Check supported DataIdentifier in active session





Number	Heading
[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
[SWS_DM_00025]	Qualified passed event using counter-based debouncing





Number	Heading
[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[32] and Autosar Diagnostic Extract Template [4] 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
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason





Number	Heading
[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
[SWS_DM_00145]	Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination





Number	Heading
[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
[SWS_DM_00230]	Check for supported subfunctions





Number	Heading
[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 <code>Active Protocol</code> in non-default session
[SWS_DM_00259]	Completion of already <code>Active Protocols</code> in default session
[SWS_DM_00260]	instances of interface <code>ClearDTC</code>
[SWS_DM_00261]	Usage of <code>ClearDTC</code> Interface
[SWS_DM_00262]	Common semantic behavior for <code>ClearDTC</code> triggered via diagnostics or application
[SWS_DM_00265]	<code>ClearDTC</code> called while another clear operation is in progress
[SWS_DM_00268]	<code>EcuReset</code> 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 <code>snapshot record</code> updates
[SWS_DM_00277]	Cancellation of <code>Active Protocol</code> in case of External Service Processing
[SWS_DM_00278]	Cancellation of <code>Active Protocol</code> in case of Internal Processing
[SWS_DM_00279]	Cancellation of <code>Active Protocol</code> before Response Transmission
[SWS_DM_00280]	Cancellation of <code>Active Protocol</code> at Response Transmission
[SWS_DM_00281]	Cancellation of active <code>DiagnosticConversation</code> in Non-Default Session
[SWS_DM_00282]	Handling of non-/active diagnostic conversations
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response





Number	Heading
[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
[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_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
[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





Number	Heading
[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 Request-Seed
[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]	





Number	Heading
[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
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[32] and Autosar Diagnostic Extract Template [4] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord





Number	Heading
[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 <code>kPrepassed</code> or <code>kPrefailed</code> for events without an assigned de-bouncing algorithm
[SWS_DM_00090]	Support of UDS service <code>ClearDiagnosticInformation</code>
[SWS_DM_00091]	Evaluation of <code>ClearDiagnosticInformation</code> parameters
[SWS_DM_00092]	Parameter range check for <code>groupOfDTC</code> 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 <code>0x14</code>
[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 <code>0x3E</code> <code>TesterPresent</code>
[SWS_DM_00127]	Availability of diagnostic service processors
[SWS_DM_00128]	Realization of UDS service <code>RequestDownload</code> (<code>0x34</code>)
[SWS_DM_00129]	Supported <code>addressAndLengthFormatIdentifier</code>
[SWS_DM_00130]	Not supported <code>addressAndLengthFormatIdentifier</code>
[SWS_DM_00131]	UDS service <code>RequestDownload</code> (<code>0x34</code>) processing
[SWS_DM_00134]	Realization of UDS service <code>RequestUpload</code> (<code>0x35</code>)





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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 <code>snapshot record</code> 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 <code>TRUE</code>
[SWS_DM_00289]	Configurable environmental condition check evaluates to <code>FALSE</code>
[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]	
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	





Number	Heading
[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[32]
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[32]
[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 <i>internal DiagnosticDataElements</i>
[SWS_DM_00397]	Retrieving data for <i>external DiagnosticDataElements</i>
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00404]	Default Service Interface for reading <i>DiagnosticDataIdentifier</i>
[SWS_DM_00407]	Default Service Interface for writing <i>DiagnosticDataIdentifier</i>
[SWS_DM_00408]	Retrieving data for requested DataIdentifier





Number	Heading
[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
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search





Number	Heading
[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 <code>Diagnostic Protocol</code>
[SWS_DM_00281]	Cancellation of active <code>DiagnosticConversation</code> 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 <code>UdsMessage</code> Pointer vendor type
[SWS_DM_00308]	Global Channel Identifier type
[SWS_DM_00316]	Header file
[SWS_DM_00317]	<code>UdsTransportProtocolHandler</code> constructor
[SWS_DM_00318]	<code>UdsTransportProtocolHandler</code> destructor
[SWS_DM_00320]	<code>UdsTransportProtocolHandler</code> <code>UdsTransportProtocolMgr</code> member
[SWS_DM_00321]	constructor member initialization
[SWS_DM_00324]	<code>UdsTransportProtocolHandler</code> <code>UdsTransportProtocolHandlerID</code> member
[SWS_DM_00328]	<code>UdsMessage</code> Pointer vendor type
[SWS_DM_00334]	<code>UdsTransportProtocolMgr</code> may be an abstract class
[SWS_DM_00335]	Header file
[SWS_DM_00339]	<code>ByteVector</code> vendor type
[SWS_DM_00402]	Reading Diagnostic Data Identifier by <code>DataIdentifier</code> interface
[SWS_DM_00403]	Reading Diagnostic Data Identifier by <code>GenericUDSService</code> interface
[SWS_DM_00405]	Writing Diagnostic Data Identifier by <code>DataIdentifier</code> interface
[SWS_DM_00406]	Writing Diagnostic Data Identifier by <code>GenericUDSService</code> interface
[SWS_DM_00410]	Check session permission
[SWS_DM_00411]	Check security level permission
[SWS_DM - CONSTR_00207]	Required <code>VINDataIdentifier</code>

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]	





Number	Heading
[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_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]	





Number	Heading
[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_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]	





Number	Heading
[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_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]	





Number	Heading
[SWS_DM_00801]	
[SWS_DM_00802]	
[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[32]





Number	Heading
[SWS_DM_00367]	No service processing
[SWS_DM_00376]	Positive response processing
[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]	





Number	Heading
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	
[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





Number	Heading
[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
[SWS_DM_00862]	Suppression of negative response for functional requests in accordance to ISO 14229-1[32]
[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





Number	Heading
[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
[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





Number	Heading
[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
[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]	





Number	Heading
[SWS_DM_00543]	
[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_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]	





Number	Heading
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[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]	





Number	Heading
[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_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]	





Number	Heading
[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]	
[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





Number	Heading
[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)
[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 <code>DiagnosticMonitor</code> service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00177]	Reaction on <code>ApplicationError</code>
[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 <code>VIN</code> 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)





Number	Heading
[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
[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[32]
[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





Number	Heading
[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]	





Number	Heading
[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_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





Number	Heading
[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 occurrence
[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]	
[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





Number	Heading
[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]	
[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]	





Number	Heading
[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
[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)





Number	Heading
[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 DDIDs in ReadDataByPeriodicIdentifier
[SWS_DM_01050]	Periodic DDID 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
[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 DDIDs
[SWS_DM_01057]	Optional stopping PDIDs after session change
[SWS_DM_01058]	Optional stopping PDIDs after security level change
[SWS_DM_01059]	No periodic DDIDs 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 DDIDs as sourceDataIdentifier
[SWS_DM_01075]	Maximum number of sourceDataIdentifiers in the request
[SWS_DM_01076]	Clearing all configured DDIDs





Number	Heading
[SWS_DM_01077]	Clearing individual configured <code>DDIDs</code>
[SWS_DM_01078]	Clear <code>DDIDs</code> on session change
[SWS_DM_01079]	Session check for <code>DDID</code>
[SWS_DM_01080]	Security level check for <code>DDID</code>
[SWS_DM_01081]	Session check for <code>sourceDataIdentifier</code>
[SWS_DM_01082]	Security level check for <code>sourceDataIdentifier</code>
[SWS_DM_01083]	Use of configured <code>DID</code> ports to get <code>DDID</code> data
[SWS_DM_CONSTR_00960]	No support for <code>DEM_AGINGCTR_UPCNT_FIRST_ACTIVE</code>

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[32] and Autosar Diagnostic Extract Template [4] of the <code>DTCFormatIdentifier</code>
[SWS_DM_00123]	Block clearing of <code>UDS DTC status byte</code> during a clear DTC operation
[SWS_DM_00124]	Limited clearing of <code>UDS DTC status byte</code> 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]	<code>UDS DTC status bit 'kConfirmedDTC'</code>
[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





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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
[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]	





Number	Heading
[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
[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 <i>event</i> 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 <i>events</i> 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.maxNumChangeOfDataIdentifierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength





Number	Heading
[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]	
[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]	





Number	Heading
[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]	





Number	Heading
[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
[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 <code>ara::diag::Authentication::VerifyCertificateUnidirectional</code> method
[SWS_DM_01232]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyCertificateUnidirectional</code> method
[SWS_DM_01233]	Successful verification of verifyCertificateUnidirectional
[SWS_DM_01234]	Unexpected verifyCertificateUnidirectional from a different client
[SWS_DM_01235]	Processing the verifyCertificateBidirectional request





Number	Heading
[SWS_DM_01236]	Handling Negative return values of <code>ara::diag::Authentication::VerifyCertificateBidirectional</code> method
[SWS_DM_01237]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyCertificateBidirectional</code> method
[SWS_DM_01238]	Successful verification of <code>verifyCertificateBidirectional</code>
[SWS_DM_01239]	Unexpected <code>verifyCertificateBidirectional</code> from a different client
[SWS_DM_01240]	Processing the <code>proofOfOwnership</code> request
[SWS_DM_01241]	Handling Negative return values of <code>ara::diag::Authentication::VerifyOwnership</code> method
[SWS_DM_01242]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyOwnership</code> method
[SWS_DM_01243]	Successful verification of Client <code>proofOfOwnership</code>
[SWS_DM_01244]	Processing the <code>deAuthenticate</code> request
[SWS_DM_01245]	Successful completion of <code>deAuthenticate</code>
[SWS_DM_01246]	Processing the <code>authenticationConfiguration</code> request
[SWS_DM_01247]	Validation of the <code>transmitCertificate</code> <code>certificateEvaluationId</code>
[SWS_DM_01248]	Processing the <code>transmitCertificate</code> request
[SWS_DM_01249]	Handling Negative return values of <code>ara::diag::Authentication::TransmitCertificate(SWS_DM_01129)</code> method
[SWS_DM_01250]	Handling unspecified negative return values of <code>ara::diag::Authentication::TransmitCertificate(SWS_DM_01129)</code> method
[SWS_DM_01251]	Successful verification of <code>transmitCertificate</code>
[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]	<code>ControlDTCSetting</code> 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





Number	Heading
[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]	
[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





Number	Heading
[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]	
[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]	





Number	Heading
[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_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]	





Number	Heading
[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]	
[SWS_DM_00639]	
[SWS_DM_00640]	





Number	Heading
[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]	
[SWS_DM_00693]	





Number	Heading
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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





Number	Heading
[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
[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 <code>ara::diag::Conversation::GetDiagnosticSecurityLevelShortName</code>
[SWS_DM_01260]	Validation of Invalid Security Level in <code>ara::diag::Conversation::GetDiagnosticSecurityLevelShortName</code>
[SWS_DM_01261]	Validation of Invalid Session Level in <code>ara::diag::Conversation::GetDiagnosticSessionShortName</code>
[SWS_DM_01262]	No storage of RoE events
[SWS_DM_01263]	Storage of RoE events
[SWS_DM_01264]	<code>DiagnosticAging.threshold</code> reached
[SWS_DM_01265]	<code>Aging</code> requires tested cycles only
[SWS_DM_01266]	Warning Indicator Request Activation
[SWS_DM_01267]	Reporting <code>kFdcThresholdReached</code> for monitor internal debouncing
[SWS_DM_01268]	Value of <code>FaultDetectionCounter</code> 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





Number	Heading
[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 <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords = 1</code>)
[SWS_DM_01277]	Triggering for <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords > 1</code>)
[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_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 <code>DTCInformation::SetNumberOfStoredEntriesNotifier</code> for suppressed DTCs
[SWS_DM_01298]	Behavior of <code>DTCInformation::SetDTCStatusChangedNotifier</code> for suppressed DTCs
[SWS_DM_01299]	Enabling of the notification <code>ara::diag::DTCInformation::SetDTCStatusChangedNotifier</code> for suppressed DTCs
[SWS_DM_01300]	Behavior of the <code>ara::diag::DTCInformation</code> class for suppressed DTCs
[SWS_DM_01301]	Behavior of <code>ara::diag::Event</code> class methods for suppressed DTCs
[SWS_DM_01302]	Behavior of <code>Diagnostic Client</code> services ClearDiagnosticInformation for suppressed DTCs inside a group
[SWS_DM_01303]	Behavior of <code>Diagnostic Client</code> 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 <code>DTCInformation::GetDtcSuppression</code>
[SWS_DM_01307]	Precondition for suppression





Number	Heading
[SWS_DM_01308]	Functionality of <code>DTCInformation::SetDtcSuppression</code>
[SWS_DM_01309]	Unblock requests after <code>ara::diag::EcuResetRequest::ExecuteReset</code> completed
[SWS_DM_01310]	Realization of UDS service <code>RequestFileTransfer</code> (0x38)
[SWS_DM_01311]	Realization of <code>modeOfOperation AddFile</code> (0x01)
[SWS_DM_01312]	Realization of <code>modeOfOperation DeleteFile</code> (0x02)
[SWS_DM_01313]	Realization of <code>modeOfOperation ReplaceFile</code> (0x03)
[SWS_DM_01314]	Realization of <code>modeOfOperation ReadFile</code> (0x04)
[SWS_DM_01315]	Realization of <code>modeOfOperation ReadDir</code> (0x05)
[SWS_DM_01316]	Realization of <code>modeOfOperation ResumeFile</code> (0x06)
[SWS_DM_01317]	Realization of <code>TransferData</code> (0x36) in context of <code>RequestFileTransfer</code>
[SWS_DM_01318]	Realization of <code>RequestTransferExit</code> (0x37) in context of <code>RequestFileTransfer</code>
[SWS_DM_01319]	Consecutive registration of notifier with <code>ReleaseHandler::SetNotifier()</code>
[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 <code>MetalInfo</code>





Number	Heading
[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()
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[SWS_DM_01452]	Mismatching versions
[SWS_DM_01453]	path to <code>version-info</code>
[SWS_DM_01454]	Query to the SOVD API version
[SWS_DM_01455]	SOVD method for SOVD API Versioning
[SWS_DM_01456]	SOVD method <code>Query an Online Capability Description</code>
[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 <code>baseTypeEncoding</code> IEEE754
[SWS_DM_01461]	SOVD data representation of <code>BITFIELD_TEXTTABLE</code>
[SWS_DM_01462]	SOVD data representation of <code>TAB_NOINTP</code>
[SWS_DM_01463]	SOVD data representation of atomic scaled numeric data with <code>texttable</code>
[SWS_DM_01464]	SOVD data representation of <code>TEXTTABLE</code>
[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]	<code>SOVD lock</code> 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]	





Number	Heading
[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]	
[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





Number	Heading
[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]	
[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 <code>environment_data</code> 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 <code>DTCFormatIdentifier</code>
[SWS_DM_00090]	Support of UDS service <code>ClearDiagnosticInformation</code>
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00126]	Realisation of UDS service <code>0x3E TesterPresent</code>
[SWS_DM_00128]	Realization of UDS service <code>RequestDownload (0x34)</code>
[SWS_DM_00134]	Realization of UDS service <code>RequestUpload (0x35)</code>
[SWS_DM_00137]	Realization of UDS service <code>TransferData (0x36)</code>
[SWS_DM_00140]	Realisation of UDS service <code>0x28 CommunicationControl</code>
[SWS_DM_00141]	Realization of UDS service <code>RequestTransferExit (0x37)</code>
[SWS_DM_00159]	Allow only to clear <code>GroupOfAllDTCs</code>
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00170]	Realisation of UDS service <code>ReadDataByIdentifier (0x22)</code>
[SWS_DM_00186]	Realisation of UDS service <code>WriteDataByIdentifier (0x2E)</code>
[SWS_DM_00201]	Realization of UDS service <code>RoutineControl (0x31)</code>
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00213]	DTC status processing
[SWS_DM_00217]	<code>UDS DTC status bit</code> transitions triggered by <code>ClearDiagnosticInformation</code> <code>UDS service</code>
[SWS_DM_00218]	<code>kConfirmedDTC</code> (Bit3) calculation
[SWS_DM_00226]	Support of UDS service <code>DiagnosticSessionControl</code>
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00229]	Support of UDS service <code>ControlDTCSetting (0x85)</code>
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00234]	Support of UDS service <code>ECUReset</code>
[SWS_DM_00235]	<code>ECUReset</code> service processing
[SWS_DM_00236]	Realization of UDS service <code>0x27 SecurityAccess</code>
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00243]	Aging-related <code>UDS DTC status byte</code> processing
[SWS_DM_00244]	Support of UDS service <code>ReadDTCInformation</code> , Subfunction <code>0x01</code>
[SWS_DM_00245]	Support of UDS service <code>ReadDTCInformation</code> , Subfunction <code>0x02</code>
[SWS_DM_00246]	Support of UDS service <code>ReadDTCInformation</code> , Subfunction <code>0x04</code>
[SWS_DM_00247]	Support of UDS service <code>ReadDTCInformation</code> , Subfunction <code>0x07</code>





Number	Heading
[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]	
[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





Number	Heading
[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
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[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]	





Number	Heading
[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]	
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00782]	





Number	Heading
[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]	
[SWS_DM_00831]	
[SWS_DM_00832]	
[SWS_DM_00833]	
[SWS_DM_00834]	
[SWS_DM_00835]	
[SWS_DM_00836]	
[SWS_DM_00837]	





Number	Heading
[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]	
[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]	





Number	Heading
[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 <code>ara::diag::EcuResetRequest::RequestReset</code> called
[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





Number	Heading
[SWS_DM_01119]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents
[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]	
[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]	





Number	Heading
[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]	





Number	Heading
[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[32]
[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[32]
[SWS_DM_00873]	Diagnostic event processing interface
[SWS_DM_01084]	Triggering for snapshot record storage (calculated)





Number	Heading
[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::DiagReportingException
[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





Number	Heading
[SWS_DM_01563]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::communicationType
[SWS_DM_01564]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::nodeIdentificationNumber
[SWS_DM_01565]	internalDataelements
[SWS_DM_01566]	
[SWS_DM_01567]	SovdFaultAttributes
[SWS_DM_01568]	supportedDolpMessageTypes
[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 Handlers
[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]	
[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





Number	Heading
[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=
[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





Number	Heading
[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
[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





Number	Heading
[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
[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::SetEventStatusChangedNotifier
[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





Number	Heading
[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
[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 WriteDataByIdentifier' 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





Number	Heading
[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
[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





Number	Heading
[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::DiagSovdErrorDomain
[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::ThrowAsException
[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::ValidateAuthorizationOutput::identity
[SWS_DM_01820]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorizationOutput::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::SovdServiceValidation
[SWS_DM_01824]	Definition of API function ara::diag::SovdServiceValidation::~~SovdServiceValidation
[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::BulkDataMetaDataDescriptor
[SWS_DM_01831]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::id
[SWS_DM_01832]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::mimetype
[SWS_DM_01833]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::name
[SWS_DM_01834]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::translation_id
[SWS_DM_01835]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::size
[SWS_DM_01836]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::file_name





Number	Heading
[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
[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=





Number	Heading
[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
[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





Number	Heading
[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::UpdatePackageStatus::phase
[SWS_DM_01904]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::status
[SWS_DM_01905]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::progress
[SWS_DM_01906]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::subprogress
[SWS_DM_01907]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step
[SWS_DM_01908]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step_translation_id
[SWS_DM_01909]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::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::GetUpdatePackageDetails
[SWS_DM_01918]	Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackageAutomated
[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::GetUpdatePackageStatus
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration
[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





Number	Heading
[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]	
[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 <i>kFailed</i>
[SWS_DM_00036]	Debounce timer behavior upon reported <i>kPassed</i>
[SWS_DM_00085]	Internal debounce timer init
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing





Number	Heading
[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::UdsTransportProtocolHandler::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
[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::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Start
[SWS_DM_00592]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Stop
[SWS_DM_00593]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::RequestResults
[SWS_DM_00594]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Offer
[SWS_DM_00596]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Read
[SWS_DM_00597]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Offer





Number	Heading
[SWS_DM_00598]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Write
[SWS_DM_00599]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Offer
[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_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
[SWS_DM_00640]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::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::SetDTCStatusChangedNotifier
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[SWS_DM_01523]	Definition of API function <code>ara::diag::DataTransferReadByPushHandler::ExitRead</code>
[SWS_DM_01546]	Definition of API function <code>ara::diag::DataTransferWriteHandler::Write</code>
[SWS_DM_01547]	Definition of API function <code>ara::diag::DataTransferWriteHandler::ExitWrite</code>

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 <code>ara::diag::Authentication::VerifyCertificateUnidirectional</code> method
[SWS_DM_01237]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyCertificateBidirectional</code> method
[SWS_DM_01242]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyOwnership</code> method
[SWS_DM_01250]	Handling unspecified negative return values of <code>ara::diag::Authentication::TransmitCertificate(SWS_DM_01129)</code> method
[SWS_DM_01438]	SOVD data attribute <code>group</code>

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





Number	Heading
[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
[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)





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00033]	Debounce timer behavior upon reported <code>kFailed</code>
[SWS_DM_00036]	Debounce timer behavior upon reported <code>kPassed</code>
[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 <code>BusyRepeatRequest</code>
[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 <code>DTCFormatIdentifier</code> in positive response <code>ReadDTCInformation.reportNumberOfDTCByStatusMask</code>
[SWS_DM_00062]	Mapping between ISO 14229-1 and Autosar Diagnostic Extract Template of the <code>DTCFormatIdentifier</code>
[SWS_DM_00063]	Providing rule for <code>DTCFormatIdentifier</code> in positive response <code>ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord</code>
[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
[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 <code>ClearDiagnosticInformation</code>
[SWS_DM_00091]	Evaluation of <code>ClearDiagnosticInformation</code> parameters
[SWS_DM_00092]	Parameter range check for <code>groupOfDTC</code> 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 <code>0x14</code>
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00116]	Clearing a <code>DTC group</code>
[SWS_DM_00117]	Clearing a DTC





Number	Heading
[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
[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)





Number	Heading
[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
[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





Number	Heading
[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 apex::diag::uds_transport::UdsMessage
[SWS_DM_00293]	Definition of API type apex::diag::uds_transport::UdsMessage::Address
[SWS_DM_00294]	Definition of API type apex::diag::uds_transport::UdsMessage::MetaInfo Map
[SWS_DM_00296]	Definition of API enum apex::diag::uds_transport::UdsMessage::Target AddressType
[SWS_DM_00297]	Definition of API function apex::diag::uds_transport::UdsMessage::GetSa
[SWS_DM_00298]	Definition of API function apex::diag::uds_transport::UdsMessage::GetTa
[SWS_DM_00299]	Definition of API function apex::diag::uds_transport::UdsMessage::GetTa Type
[SWS_DM_00300]	Definition of API function apex::diag::uds_transport::UdsMessage::Get Payload
[SWS_DM_00301]	Definition of API function apex::diag::uds_transport::UdsMessage::Get Payload
[SWS_DM_00302]	Definition of API function apex::diag::uds_transport::UdsMessage::AddMeta Info
[SWS_DM_00303]	Definition of API type apex::diag::uds_transport::UdsMessagePtr
[SWS_DM_00304]	Definition of API type apex::diag::uds_transport::UdsMessageConstPtr
[SWS_DM_00306]	Definition of API class apex::diag::uds_transport::UdsTransportProtocolMgr
[SWS_DM_00307]	Definition of API enum apex::diag::uds_transport::UdsTransportProtocol Mgr::TransmissionResult
[SWS_DM_00309]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::IndicateMessage
[SWS_DM_00310]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::NotifyMessageFailure
[SWS_DM_00311]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::HandleMessage
[SWS_DM_00312]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::TransmitConfirmation
[SWS_DM_00313]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::ChannelReestablished
[SWS_DM_00314]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Mgr::HandlerStopped
[SWS_DM_00315]	Definition of API class apex::diag::uds_transport::UdsTransportProtocol Handler
[SWS_DM_00319]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Handler::Initialize
[SWS_DM_00322]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Handler::Start





Number	Heading
[SWS_DM_00323]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::Stop
[SWS_DM_00325]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::GetHandlerID
[SWS_DM_00326]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment
[SWS_DM_00327]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::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 apex::diag::uds_transport::UdsTransportProtocolHandlerID
[SWS_DM_00337]	Definition of API type apex::diag::uds_transport::ChannelID
[SWS_DM_00338]	Definition of API type apex::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
[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





Number	Heading
[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 <code>apext::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult</code>
[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) resources
[SWS_DM_00388]	Filling provided <code>UdsMessage</code>
[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 <code>UdsMessage</code>
[SWS_DM_00393]	Retrieving data for <code>internal DiagnosticDataElements</code>
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00409]	Check supported <code>DataIdentifier</code>
[SWS_DM_00412]	Check requested number of <code>DataIdentifiers</code>
[SWS_DM_00413]	Check supported <code>DataIdentifier</code> in active session
[SWS_DM_00414]	Check supported <code>DataIdentifier</code> on active security level
[SWS_DM_00415]	Check supported <code>DataIdentifier</code>
[SWS_DM_00416]	Check supported <code>DataIdentifier</code> in active session
[SWS_DM_00417]	Check supported <code>DataIdentifier</code> 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 <code>RoutineIdentifier</code>
[SWS_DM_00448]	Check supported <code>RoutineIdentifier</code> 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





Number	Heading
[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
[SWS_DM_00527]	Definition of API class ara::diag::DiagUdsNrcException
[SWS_DM_00528]	Definition of API function ara::diag::DiagUdsNrcException::DiagUdsNrcException
[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::DiagUdsNrcErrorDomain
[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::ThrowAsException
[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





Number	Heading
[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
[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





Number	Heading
[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>}
[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





Number	Heading
[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
[SWS_DM_00615]	Definition of API function ara::diag::CancellationHandler::SetNotifier
[SWS_DM_00616]	Definition of API function ara::diag::GenericUDSService::GenericUDSService
[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





Number	Heading
[SWS_DM_00631]	Definition of API variable ara::diag::GenericDataIdentifier::OperationOutput::responseData
[SWS_DM_00632]	Definition of API variable ara::diag::GenericUDSService::OperationOutput::responseData
[SWS_DM_00633]	Definition of API variable ara::diag::GenericRoutine::OperationOutput::responseData
[SWS_DM_00634]	Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier
[SWS_DM_00635]	Definition of API function ara::diag::GenericDataIdentifier::~GenericDataIdentifier
[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
[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::SnapshotDataIdentifierType
[SWS_DM_00661]	Definition of API class ara::diag::DTCInformation::SnapshotDataRecordType
[SWS_DM_00662]	Definition of API class ara::diag::DTCInformation::SnapshotRecordUpdatedType





Number	Heading
[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::SetDTCStatusChangedNotifier
[SWS_DM_00668]	Definition of API function ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier
[SWS_DM_00669]	Definition of API function ara::diag::DTCInformation::GetNumberOfStoredEntries
[SWS_DM_00670]	Definition of API function ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier
[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::SetControlDtcStatusNotifier
[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
[SWS_DM_00696]	Definition of API function ara::diag::Conversation::GetDiagnosticSession
[SWS_DM_00697]	Definition of API function ara::diag::Conversation::SetDiagnosticSessionNotifier
[SWS_DM_00698]	Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevel
[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::GetDiagnosticSessionShortName
[SWS_DM_00708]	Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevelShortName
[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





Number	Heading
[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::DoIPGroupIdentification
[SWS_DM_00723]	Definition of API function ara::diag::DoIPGroupIdentification::~DoIPGroupIdentification
[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
[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





Number	Heading
[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
[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





Number	Heading
[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::GetDoIPTriggerVehicleAnnouncement
[SWS_DM_00822]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::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::GetNetworkInterfaceId
[SWS_DM_00834]	Definition of API function ara::diag::DoIPActivationLine::UpdateActivationLineState
[SWS_DM_00835]	Definition of API function ara::diag::DoIPActivationLine::GetActivationLineState
[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





Number	Heading
[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 <code>kFinallyDefective</code>
[SWS_DM_00878]	Starting time-based event debouncing towards <code>kFinallyHealed</code>
[SWS_DM_00880]	Debounce time freeze request
[SWS_DM_00882]	Enable condition influence on debouncing behavior
[SWS_DM_00883]	<code>UDS DTC status bit</code> transitions triggered by test results
[SWS_DM_00886]	Observability of the <code>Event status</code> byte
[SWS_DM_00888]	Observability of indicator status
[SWS_DM_00894]	Notification event upon <code>snapshot record</code> updates
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00896]	Handling of <code>DiagnosticClearConditions</code>
[SWS_DM_00897]	Usage of ClearDTC Interface
[SWS_DM_00898]	ClearDTC call on invalid DTC or <code>DTC group</code>
[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]	<code>NumberOfStoredEntries</code>
[SWS_DM_00905]	Retrieving data for <code>external DiagnosticDataElements</code>
[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 <code>ara::diag::DTCInformation::SetEventMemoryOverflowNotifier</code>
[SWS_DM_00919]	Definition of API function <code>ara::diag::DTCInformation::GetEventMemoryOverflow</code>
[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





Number	Heading
[SWS_DM_00923]	Event memory overflow set condition
[SWS_DM_00924]	Event memory overflow reset condition
[SWS_DM_00925]	Event memory overflow notifier on occurrence
[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
[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





Number	Heading
[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
[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::DiagReportingException
[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::DiagOfferErrorDomain
[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::ThrowAsException





Number	Heading
[SWS_DM_00996]	Definition of API class <code>ara::diag::DiagReportingErrorDomain</code>
[SWS_DM_00997]	Definition of API type <code>ara::diag::DiagReportingErrorDomain::Errc</code>
[SWS_DM_00998]	Definition of API type <code>ara::diag::DiagReportingErrorDomain::Exception</code>
[SWS_DM_00999]	Definition of API function <code>ara::diag::DiagReportingErrorDomain::DiagReportingErrorDomain</code>
[SWS_DM_01000]	Definition of API function <code>ara::diag::DiagReportingErrorDomain::Name</code>
[SWS_DM_01001]	Definition of API function <code>ara::diag::DiagReportingErrorDomain::Message</code>
[SWS_DM_01002]	Definition of API function <code>ara::diag::DiagReportingErrorDomain::ThrowAsException</code>
[SWS_DM_01003]	Definition of API function <code>ara::diag::GetDiagOfferDomain</code>
[SWS_DM_01004]	Definition of API function <code>ara::diag::GetDiagReportingDomain</code>
[SWS_DM_01005]	Definition of API function <code>ara::diag::MakeErrorCode</code>
[SWS_DM_01006]	Definition of API function <code>ara::diag::MakeErrorCode</code>
[SWS_DM_01007]	Definition of API type <code>ara::diag::ResetRequestType</code>
[SWS_DM_01009]	Definition of API class <code>ara::diag::EcuResetRequest</code>
[SWS_DM_01010]	Definition of API function <code>ara::diag::EcuResetRequest::EcuResetRequest</code>
[SWS_DM_01011]	Definition of API function <code>ara::diag::EcuResetRequest::~~EcuResetRequest</code>
[SWS_DM_01012]	Definition of API function <code>ara::diag::EcuResetRequest::EnableRapidShutdown</code>
[SWS_DM_01013]	Definition of API function <code>ara::diag::EcuResetRequest::RequestReset</code>
[SWS_DM_01014]	Definition of API function <code>ara::diag::EcuResetRequest::ExecuteReset</code>
[SWS_DM_01016]	Definition of API function <code>ara::diag::EcuResetRequest::Offer</code>
[SWS_DM_01017]	Definition of API function <code>ara::diag::EcuResetRequest::StopOffer</code>
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after <code>ara::diag::EcuResetRequest::RequestReset</code> called
[SWS_DM_01023]	Calling <code>ExecuteReset()</code> if positive response shall be sent before reset
[SWS_DM_01024]	Event Status processing
[SWS_DM_01025]	<code>Event status</code> bit transitions triggered by test results
[SWS_DM_01026]	Resetting the status of an <code>Event</code>
[SWS_DM_01027]	<code>Event status</code> bit transitions triggered by <code>operation cycle</code> restarting
[SWS_DM_01028]	<code>Event status</code> bit transitions triggered by <code>ClearDiagnosticInformation UDS service</code>
[SWS_DM_01029]	Notification about <code>Event status</code> changes
[SWS_DM_01030]	Observability of the <code>UDS DTC status byte</code>
[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





Number	Heading
[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
[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::UdsTransportProtocolPeriodicHandler
[SWS_DM_01065]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages
[SWS_DM_01066]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength
[SWS_DM_01067]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit





Number	Heading
[SWS_DM_01068]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::GetPeriodicHandler
[SWS_DM_01069]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation
[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_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
[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 ControlDTCSetting 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





Number	Heading
[SWS_DM_01106]	Applicability of Event Combination
[SWS_DM_01107]	DTC Status Byte calculation
[SWS_DM_01108]	Clear all DTCs <i>event</i> 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 <i>events</i> 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.maxNumChangeOfDataIdentifierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisonOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength
[SWS_DM_01122]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumberOfStoredDTCStatusChangedEvents
[SWS_DM_01123]	Definition of API class ara::diag::Authentication
[SWS_DM_01124]	Definition of API function ara::diag::Authentication::Authentication
[SWS_DM_01125]	Definition of API function ara::diag::Authentication::~~Authentication
[SWS_DM_01126]	Definition of API function ara::diag::Authentication::VerifyCertificateUnidirectional
[SWS_DM_01127]	Definition of API function ara::diag::Authentication::VerifyCertificateBidirectional
[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::~~ClientAuthentication
[SWS_DM_01137]	Definition of API function ara::diag::ClientAuthentication::ClientAuthentication
[SWS_DM_01138]	Definition of API function ara::diag::ClientAuthentication::operator=





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[SWS_DM_01262]	No storage of RoE events
[SWS_DM_01263]	Storage of RoE events
[SWS_DM_01264]	<code>DiagnosticAging.threshold</code> reached
[SWS_DM_01265]	<code>Aging</code> requires tested cycles only
[SWS_DM_01266]	Warning Indicator Request Activation
[SWS_DM_01267]	Reporting <code>kFdcThresholdReached</code> for monitor internal debouncing
[SWS_DM_01268]	Value of <code>FaultDetectionCounter</code> 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 <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords</code> = 1)
[SWS_DM_01277]	Triggering for <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords</code> > 1)
[SWS_DM_01278]	Definition of API variable <code>ara::diag::kDefaultSession</code>
[SWS_DM_01279]	Definition of API variable <code>ara::diag::kProgrammingSession</code>
[SWS_DM_01280]	Definition of API variable <code>ara::diag::kExtendedDiagnosticSession</code>
[SWS_DM_01281]	Definition of API variable <code>ara::diag::kSafetySystemDiagnosticSession</code>
[SWS_DM_01282]	Definition of API variable <code>ara::diag::kLocked</code>
[SWS_DM_01283]	Definition of API variable <code>ara::diag::kSoftReset</code>
[SWS_DM_01284]	Definition of API variable <code>ara::diag::kHardReset</code>
[SWS_DM_01285]	Definition of API variable <code>ara::diag::kKeyOffOnReset</code>
[SWS_DM_01286]	Definition of API variable <code>ara::diag::kCustomReset</code>
[SWS_DM_01309]	Unblock requests after <code>ara::diag::EcuResetRequest::ExecuteReset</code> 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 <code>ReleaseHandler::SetNotifier()</code>
[SWS_DM_01320]	Definition of API class <code>ara::diag::FileTransferService</code>





Number	Heading
[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
[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::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()
[SWS_DM_01354]	Consecutive registration of notifier with SetEventMemoryOverflowNotifier()





Number	Heading
[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::DoIPEntityIdentification
[SWS_DM_01365]	Definition of API function ara::diag::DoIPEntityIdentification::~~DoIPEntityIdentification
[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
[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





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_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
[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





Number	Heading
[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
[SWS_DM_01458]	SOVD data representation of units
[SWS_DM_01459]	SOVD data representation of textual Strings
[SWS_DM_01460]	SOVD data representation of <code>baseTypeEncoding</code> IEEE754
[SWS_DM_01461]	SOVD data representation of <code>BITFIELD_TEXTTABLE</code>
[SWS_DM_01462]	SOVD data representation of <code>TAB_NOINTP</code>
[SWS_DM_01463]	SOVD data representation of atomic scaled numeric data with texttable
[SWS_DM_01464]	SOVD data representation of <code>TEXTTABLE</code>
[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





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_01478]	Definition of API function ara::diag::SovdProximityChallenge::Offer
[SWS_DM_01479]	Definition of API function ara::diag::SovdProximityChallenge::~SovdProximityChallenge
[SWS_DM_01480]	Definition of API function ara::diag::SovdProximityChallenge::SovdProximityChallenge
[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::ValidateResponse
[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::entityIdentification
[SWS_DM_01497]	Definition of API class ara::diag::DataTransferReadSharedDataHandler
[SWS_DM_01498]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler
[SWS_DM_01499]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::~DataTransferReadSharedDataHandler
[SWS_DM_01500]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler
[SWS_DM_01501]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler





Number	Heading
[SWS_DM_01502]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::operator=
[SWS_DM_01503]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::operator=
[SWS_DM_01504]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::Read
[SWS_DM_01505]	Definition of API function ara::diag::DataTransferReadSharedDataHandler::ExitRead
[SWS_DM_01506]	Definition of API class ara::diag::DataTransferReadByPullHandler
[SWS_DM_01507]	Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler
[SWS_DM_01508]	Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler
[SWS_DM_01509]	Definition of API function ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler
[SWS_DM_01510]	Definition of API function ara::diag::DataTransferReadByPullHandler::~DataTransferReadByPullHandler
[SWS_DM_01511]	Definition of API function ara::diag::DataTransferReadByPullHandler::operator=
[SWS_DM_01512]	Definition of API function ara::diag::DataTransferReadByPullHandler::operator=
[SWS_DM_01513]	Definition of API function ara::diag::DataTransferReadByPullHandler::Read
[SWS_DM_01514]	Definition of API function ara::diag::DataTransferReadByPullHandler::ExitRead
[SWS_DM_01515]	Definition of API class ara::diag::DataTransferReadByPushHandler
[SWS_DM_01516]	Definition of API function ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler
[SWS_DM_01517]	Definition of API function ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler
[SWS_DM_01518]	Definition of API function ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler
[SWS_DM_01519]	Definition of API function ara::diag::DataTransferReadByPushHandler::~DataTransferReadByPushHandler
[SWS_DM_01520]	Definition of API function ara::diag::DataTransferReadByPushHandler::operator=
[SWS_DM_01521]	Definition of API function ara::diag::DataTransferReadByPushHandler::operator=
[SWS_DM_01522]	Definition of API function ara::diag::DataTransferReadByPushHandler::Read
[SWS_DM_01523]	Definition of API function ara::diag::DataTransferReadByPushHandler::ExitRead
[SWS_DM_01524]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::~UdsTransportProtocolMgr
[SWS_DM_01525]	ara::diag::DoIPPowerMode not yet offered when client requests DoIP PowerMode





Number	Heading
[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::DataTransferWriteHandler
[SWS_DM_01541]	Definition of API function ara::diag::DataTransferWriteHandler::~~DataTransferWriteHandler
[SWS_DM_01542]	Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler
[SWS_DM_01543]	Definition of API function ara::diag::DataTransferWriteHandler::operator=
[SWS_DM_01544]	Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler
[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
[SWS_DM_01548]	Definition of API class ara::diag::DataTransferReadSession
[SWS_DM_01549]	Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession
[SWS_DM_01550]	Definition of API function ara::diag::DataTransferReadSession::~~DataTransferReadSession
[SWS_DM_01551]	Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession
[SWS_DM_01552]	Definition of API function ara::diag::DataTransferReadSession::operator=
[SWS_DM_01553]	Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession
[SWS_DM_01554]	Definition of API function ara::diag::DataTransferReadSession::operator=
[SWS_DM_01555]	Definition of API class ara::diag::DataTransferWriteSession





Number	Heading
[SWS_DM_01556]	Definition of API function <code>ara::diag::DataTransferWriteSession::~DataTransferWriteSession</code>
[SWS_DM_01557]	Definition of API function <code>ara::diag::DataTransferWriteSession::DataTransferWriteSession</code>
[SWS_DM_01558]	Definition of API function <code>ara::diag::DataTransferWriteSession::operator=</code>
[SWS_DM_01559]	Definition of API function <code>ara::diag::DataTransferWriteSession::DataTransferWriteSession</code>
[SWS_DM_01560]	Definition of API function <code>ara::diag::DataTransferWriteSession::operator=</code>
[SWS_DM_01561]	Response Body for SOVD fault <code>environment_data</code> table
[SWS_DM_01562]	Definition of API variable <code>ara::diag::CommunicationControl::ComCtrlRequestParamsType::controlType</code>
[SWS_DM_01563]	Definition of API variable <code>ara::diag::CommunicationControl::ComCtrlRequestParamsType::communicationType</code>
[SWS_DM_01564]	Definition of API variable <code>ara::diag::CommunicationControl::ComCtrlRequestParamsType::nodeIdentificationNumber</code>
[SWS_DM_01565]	Supported <code>internal DiagnosticDataElements</code>
[SWS_DM_01566]	Subfunctions of 0x19 / ReadDTCInformation with chronological reporting order
[SWS_DM_01567]	Response Body for SOVD fault attributes
[SWS_DM_01568]	<code>supportedDolpMessageTypes</code>
[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 <code>Transport Protocol Handlers</code>
[SWS_DM_01574]	Write data to be persisted
[SWS_DM_01576]	Behavior of not configured <code>DiagnosticMemoryDestination.agingRequiresTestedCycle</code>
[SWS_DM_01577]	Behavior of not configured <code>DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear</code>
[SWS_DM_01578]	Behavior of not configured <code>DiagnosticMemoryDestination.clearDtcLimitation</code>
[SWS_DM_01579]	Behavior of not configured <code>DiagnosticMemoryDestination.memoryEntryStorageTrigger</code>
[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 <code>ara::diag::OperationCycle::operator=</code>
[SWS_DM_01597]	Definition of API function <code>ara::diag::OperationCycle::operator=</code>
[SWS_DM_01598]	Definition of API function <code>ara::diag::OperationCycle::OperationCycle</code>





Number	Heading
[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=
[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





Number	Heading
[SWS_DM_01643]	Definition of API function <code>ara::diag::DoIPPowerMode::operator=</code>
[SWS_DM_01644]	Definition of API function <code>ara::diag::DoIPPowerMode::operator=</code>
[SWS_DM_01645]	Definition of API function <code>ara::diag::DoIPPowerMode::DoIPPowerMode</code>
[SWS_DM_01646]	Definition of API function <code>ara::diag::DoIPPowerMode::DoIPPowerMode</code>
[SWS_DM_01647]	Definition of API function <code>ara::diag::DownloadService::operator=</code>
[SWS_DM_01648]	Definition of API function <code>ara::diag::DownloadService::operator=</code>
[SWS_DM_01649]	Definition of API function <code>ara::diag::DownloadService::DownloadService</code>
[SWS_DM_01650]	Definition of API function <code>ara::diag::DownloadService::DownloadService</code>
[SWS_DM_01651]	Definition of API function <code>ara::diag::GenericDataIdentifier::operator=</code>
[SWS_DM_01652]	Definition of API function <code>ara::diag::GenericDataIdentifier::operator=</code>
[SWS_DM_01653]	Definition of API function <code>ara::diag::GenericDataIdentifier::GenericDataIdentifier</code>
[SWS_DM_01654]	Definition of API function <code>ara::diag::GenericDataIdentifier::GenericDataIdentifier</code>
[SWS_DM_01655]	Definition of API function <code>ara::diag::GenericUDSService::operator=</code>
[SWS_DM_01656]	Definition of API function <code>ara::diag::GenericUDSService::operator=</code>
[SWS_DM_01657]	Definition of API function <code>ara::diag::GenericUDSService::GenericUDSService</code>
[SWS_DM_01658]	Definition of API function <code>ara::diag::GenericUDSService::GenericUDSService</code>
[SWS_DM_01659]	Definition of API function <code>{<namespace-list-data-element>>::{<data-element-interface-name>}::operator=</code>
[SWS_DM_01660]	Definition of API function <code>{<namespace-list-data-element>>::{<data-element-interface-name>}::operator=</code>
[SWS_DM_01661]	Definition of API function <code>{<namespace-list-data-element>>::{<data-element-interface-name>}::<data-element-interface-name></code>
[SWS_DM_01662]	Definition of API function <code>{<namespace-list-data-element>>::{<data-element-interface-name>}::<data-element-interface-name></code>
[SWS_DM_01663]	Definition of API function <code>{<namespace-list-data-identifier>>::{<data-identifier-interface-name>}::operator=</code>
[SWS_DM_01664]	Definition of API function <code>{<namespace-list-data-identifier>>::{<data-identifier-interface-name>}::operator=</code>
[SWS_DM_01665]	Definition of API function <code>{<namespace-list-data-identifier>>::{<data-identifier-interface-name>}::<data-identifier-interface-name></code>
[SWS_DM_01666]	Definition of API function <code>{<namespace-list-data-identifier>>::{<data-identifier-interface-name>}::<data-identifier-interface-name></code>
[SWS_DM_01667]	Definition of API function <code>{<routine-interface-hierarchical-namespace-list>>::{<routine-interface-name>}::operator=</code>





Number	Heading
[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::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
[SWS_DM_01702]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor





Number	Heading
[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::SetEventStatusChangedNotifier
[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
[SWS_DM_01755]	Definition of API function ara::diag::EventStatusByte::IsSet





Number	Heading
[SWS_DM_01756]	Definition of API function <code>ara::diag::EventStatusByte::IsPassedAndTested</code>
[SWS_DM_01757]	Definition of API function <code>ara::diag::EventStatusByte::IsFailedAndTested</code>
[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
[SWS_DM_01804]	Realization of SOVD Register an Update at the SOVD Server
[SWS_DM_01806]	Definition of API enum <code>ara::diag::DiagSovdErrc</code>
[SWS_DM_01807]	Definition of API class <code>ara::diag::DiagSovdException</code>
[SWS_DM_01808]	Definition of API function <code>ara::diag::DiagSovdException::DiagSovdException</code>
[SWS_DM_01809]	Definition of API class <code>ara::diag::DiagSovdErrorDomain</code>
[SWS_DM_01810]	Definition of API type <code>ara::diag::DiagSovdErrorDomain::Errc</code>
[SWS_DM_01811]	Definition of API type <code>ara::diag::DiagSovdErrorDomain::Exception</code>
[SWS_DM_01812]	Definition of API function <code>ara::diag::DiagSovdErrorDomain::DiagSovdErrorDomain</code>





Number	Heading
[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::ThrowAsException
[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::ValidateAuthorizationOutput::identity
[SWS_DM_01820]	Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorizationOutput::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::SovdServiceValidation
[SWS_DM_01824]	Definition of API function ara::diag::SovdServiceValidation::~~SovdServiceValidation
[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::BulkDataMetaDataDescriptor
[SWS_DM_01831]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::id
[SWS_DM_01832]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::mimetype
[SWS_DM_01833]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::name
[SWS_DM_01834]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::translation_id
[SWS_DM_01835]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::size
[SWS_DM_01836]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::file_name
[SWS_DM_01837]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::creation_date
[SWS_DM_01838]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::last_modified
[SWS_DM_01839]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::hash





Number	Heading
[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
[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





Number	Heading
[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
[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





Number	Heading
[SWS_DM_01902]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageStatus
[SWS_DM_01903]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::phase
[SWS_DM_01904]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::status
[SWS_DM_01905]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::progress
[SWS_DM_01906]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::subprogress
[SWS_DM_01907]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step
[SWS_DM_01908]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::step_translation_id
[SWS_DM_01909]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::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::GetUpdatePackageDetails
[SWS_DM_01918]	Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackageAutomated
[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::GetUpdatePackageStatus
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration
[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::ValidateAuthorizationOutput
[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





Number	Heading
[SWS_DM_09010]	Definition of API function apex::diag::uds_transport::UdsMessage::~Uds Message
[SWS_DM_09012]	Definition of API function apex::diag::uds_transport::UdsMessage::Uds Message
[SWS_DM_09015]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Handler::UdsTransportProtocolHandler
[SWS_DM_09016]	Definition of API function apex::diag::uds_transport::UdsTransportProtocol Handler::~UdsTransportProtocolHandler
[SWS_DM_09017]	Definition of API enum apex::diag::uds_transport::UdsTransportProtocol Handler::InitializationResult
[SWS_DM_09021]	Definition of API type apex::diag::uds_transport::UdsTransportProtocol Mgr::GlobalChannelIdentifier
[SWS_DM_09025]	Definition of API variable apex::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
[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





Number	Heading
[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
[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





Number	Heading
[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 WriteDataByIdentifier' 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 <code>SOVD ara::diag</code> 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]	
[SWS_DM_01939]	
[SWS_DM_01940]	
[SWS_DM_01941]	
[SWS_DM_01942]	
[SWS_DM_01943]	
[SWS_DM_01944]	
[SWS_DM_01945]	





Number	Heading
[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
[SWS_DM_CONSTR_00206]	Supported format for data identifier for VINDataIdentifier
[SWS_DM_CONSTR_00394]	<i>Internal DiagnosticDataElements</i> are read-only
[SWS_DM_CONSTR_00395]	Restriction on DEM-exclusive <i>DiagnosticDataElements</i>





Number	Heading
[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

E.11 Constraint and Specification Item History of this document according to AUTOSAR Release R25-11

E.11.1 Added Specification Items in R25-11

Number	Heading
[SWS_DM_02061]	ViolationMessage InvalidDebouncingAlgorithmViolation
[SWS_DM_02062]	ViolationMessage UnexpectedMonitorActionHandlingViolation
[SWS_DM_02090]	Definition of API function <code>ara::diag::Conversation::Conversation</code>
[SWS_DM_02091]	Definition of API function <code>ara::diag::Conversation::operator=</code>
[SWS_DM_02092]	Definition of API function <code>ara::diag::Conversation::Conversation</code>
[SWS_DM_02093]	Definition of API function <code>ara::diag::Conversation::operator=</code>
[SWS_DM_02094]	Provision of source address
[SWS_DM_02095]	Provision of target address
[SWS_DM_02096]	Provision of UDS addressing mode
[SWS_DM_02097]	Provision of request handle
[SWS_DM_02098]	Provision of local IP address
[SWS_DM_02099]	Provision of local port number
[SWS_DM_02102]	Provision of remote IP address
[SWS_DM_02103]	Provision of remote port number
[SWS_DM_02104]	Provision of DTC number
[SWS_DM_02105]	Provision of kDoIP context





Number	Heading
[SWS_DM_02106]	Valid values for the attribute <code>maxTesterConnections</code> of meta-class <code>DoIpNetworkConfiguration</code>
[SWS_DM_02107]	Vehicle Announcement/Identification Response Message without "VIN/GID sync status" byte
[SWS_DM_02108]	Default value for the attribute <code>sourceAddressRangeStart</code> of meta-class <code>GenericTpConnection</code>
[SWS_DM_02109]	Default value for the attribute <code>sourceAddressRangeEnd</code> of meta-class <code>GenericTpConnection</code>
[SWS_DM_02110]	Default value for the attribute <code>priority</code> of meta-class <code>GenericTpConnection</code>
[SWS_DM_02111]	DoIP protocol semantics based on the configured DoIP protocol version
[SWS_DM_02112]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotDataIdentifierType::dataIdentifier</code>
[SWS_DM_02113]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotDataIdentifierType::data</code>
[SWS_DM_02114]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotDataRecordType::snapshotRecordNumber</code>
[SWS_DM_02115]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotDataRecordType::snapshotDataIdentifiers</code>
[SWS_DM_02116]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotRecordUpdatedType::DTC</code>
[SWS_DM_02117]	Definition of API variable <code>ara::diag::DTCInformation::SnapshotRecordUpdatedType::snapshotDataRecords</code>
[SWS_DM_02118]	Definition of API type <code>ara::diag::EventStatusByteNotifier</code>
[SWS_DM_02119]	Response Body for SOVD fault <code>environment_data</code>
[SWS_DM_02120]	Response Body for SOVD fault attributes
[SWS_DM_02121]	Definition of API type <code>ara::diag::InitMonitorNotifier</code>
[SWS_DM_02122]	Definition of API type <code>ara::diag::DTCInformation::DtcStatusChangedNotifier</code>
[SWS_DM_02123]	Definition of API type <code>ara::diag::DTCInformation::SnapshotRecordUpdatedTypeNotifier</code>
[SWS_DM_02124]	Definition of API type <code>ara::diag::DTCInformation::NumberOfStoredEntriesNotifier</code>
[SWS_DM_02125]	Definition of API type <code>ara::diag::DTCInformation::ControlDtcStatusTypeNotifier</code>
[SWS_DM_02126]	Definition of API type <code>ara::diag::Conversation::ActivityStatusTypeNotifier</code>
[SWS_DM_02127]	Definition of API type <code>ara::diag::Conversation::SessionControlTypeNotifier</code>
[SWS_DM_02128]	Definition of API type <code>ara::diag::Conversation::SecurityLevelTypeNotifier</code>
[SWS_DM_02129]	Definition of API type <code>ara::diag::Indicator::IndicatorStatusTypeNotifier</code>
[SWS_DM_02130]	Access control on <code>ara::diag</code> interfaces
[SWS_DM_02131]	Security event for access control
[SWS_DM_02132]	Reporting DM access control denied to <code>IdsM</code>
[SWS_DM_02133]	Security event context data definition: <code>SEV_ACCESS_CONTROL_DM_IAM_ACCESS_DENIED</code>





Number	Heading
[SWS_DM_02134]	Reporting "DoIP header check failed" to IdsM
[SWS_DM_02135]	Security event context data definition: SEV_DOIP_HEADER_CHECK_FAILED
[SWS_DM_02136]	Reporting "DoIP routing request rejected" to IdsM
[SWS_DM_02137]	Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_CHECK_FAILED
[SWS_DM_02138]	Reporting "DoIP routing request accepted" to IdsM
[SWS_DM_02139]	Security event context data definition: SEV_DOIP_ROUTING_ACTIVATION_SUCCESS
[SWS_DM_02140]	Reporting "DoIP diagnostic message check failed" to IdsM
[SWS_DM_02141]	Security event context data definition: SEV_DOIP_DIAG_MESSAGE_CHECK_FAILED
[SWS_DM_02142]	SOVD Snapshot record layout
[SWS_DM_02145]	Support of <code>internal</code> data for ExtendDataRecords
[SWS_DM_02146]	Definition of API class { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x></code> }
[SWS_DM_02147]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::</code> { <code><extended-data-record-element-interface-name-2></code> }
[SWS_DM_02148]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::operator=</code>
[SWS_DM_02149]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::</code> { <code><extended-data-record-element-interface-name-4></code> }
[SWS_DM_02150]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::operator=</code>
[SWS_DM_02151]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::~</code> { <code><extended-data-record-element-interface-name-6></code> }
[SWS_DM_02152]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::Offer</code>
[SWS_DM_02153]	Definition of API function { <code><namespace-list-extended-data-record-element>::</code> { <code><extended-data-record-element-interface-name-x>::StopOffer</code>





Number	Heading
[SWS_DM_02154]	Definition of API class { <namespace-list-extended-data-record-element>>::{ <extended-data-record-element-interface-name-x>>::{ <extended-data-record-element-name-upper-camel>} Read Output
[SWS_DM_02155]	Definition of API variable { <namespace-list-extended-data-record-element>>::{ <extended-data-record-element-interface-name-x>>::{ <extended-data-record-element-out-arg-symbol>}
[SWS_DM_02156]	Definition of API function { <namespace-list-extended-data-record-element>>::{ <extended-data-record-element-interface-name-x>}::Read
[SWS_DM_02157]	Diagnostic ExtendedDataRecordElement Header File: file name, includes and multiple inclusion guard
[SWS_DM_02158]	Definition of API function { <namespace-list-extended-data-record-element>>::{ <extended-data-record-element-interface-name-x>>::{ <extended-data-record-element-interface-name-1>}
[SWS_DM_02159]	Diagnostic ExtendedDataRecordElement Header File: service namespace
[SWS_DM_02160]	Retrieve resource Data groups
[SWS_DM_02161]	Retrieval of SOVD entity data
[SWS_DM_02162]	SOVD method Query for data from an Entity with categories filter
[SWS_DM_02163]	SOVD method Query for data from an Entity with group filter
[SWS_DM_02164]	Retrieval of SOVD content
[SWS_DM_02165]	Retrieval of SOVD content element data
[SWS_DM_02166]	SOVD method Write a data resource from an Entity – data query
[SWS_DM_02167]	Retrieval of SOVD entity operations
[SWS_DM_02170]	Retrieval of SOVD entity executions of an operation
[SWS_DM_02171]	Operation execution without required proximity_challenge
[SWS_DM_02172]	Operation execution with wrong proximity_challenge
[SWS_DM_02173]	Operation execution with successful proximity_challenge
[SWS_DM_02174]	Operation start execution
[SWS_DM_02175]	Retrieval of SOVD details of an operation
[SWS_DM_02176]	Response body of a synchronous operation start
[SWS_DM_02177]	Response body of an asynchronous operation start
[SWS_DM_02178]	Support of SOVD method control of operations from an Entity
[SWS_DM_02179]	Operation terminate execution
[SWS_DM_02180]	Operation Read the Status of an operation
[SWS_DM_02181]	Operation status
[SWS_DM_02182]	Definition of API class ara::diag::SovdContentConcurrencyType





Number	Heading
[SWS_DM_02183]	Definition of API variable ara::diag::SovdContentConcurrencyType::read
[SWS_DM_02184]	Definition of API variable ara::diag::SovdContentConcurrencyType::write
[SWS_DM_02185]	Definition of API variable ara::diag::SovdContentConcurrencyType::read Write
[SWS_DM_02186]	Definition of API class ara::diag::GenericDiagnosticSovdContent
[SWS_DM_02187]	Definition of API function ara::diag::GenericDiagnosticSovdContent::GenericDiagnosticSovdContent
[SWS_DM_02188]	Definition of API function ara::diag::GenericDiagnosticSovdContent::~~GenericDiagnosticSovdContent
[SWS_DM_02189]	Definition of API function ara::diag::GenericDiagnosticSovdContent::GenericDiagnosticSovdContent
[SWS_DM_02190]	Definition of API function ara::diag::GenericDiagnosticSovdContent::operator=
[SWS_DM_02191]	Definition of API function ara::diag::GenericDiagnosticSovdContent::Read
[SWS_DM_02192]	Definition of API function ara::diag::GenericDiagnosticSovdContent::Write
[SWS_DM_02193]	Definition of API function ara::diag::GenericDiagnosticSovdContent::Offer
[SWS_DM_02194]	Definition of API function ara::diag::GenericDiagnosticSovdContent::Stop Offer
[SWS_DM_02195]	Definition of API enum ara::diag::OperationExecutionStatus
[SWS_DM_02196]	Definition of API enum ara::diag::TerminateOperationExecutionReason
[SWS_DM_02197]	Definition of API class ara::diag::SovdOperationExecution
[SWS_DM_02198]	Definition of API function ara::diag::SovdOperationExecution::SovdOperationExecution
[SWS_DM_02199]	Definition of API function ara::diag::SovdOperationExecution::~~SovdOperationExecution
[SWS_DM_02200]	Definition of API class ara::diag::SovdOperationExecution::GetStatus Response
[SWS_DM_02201]	Definition of API function ara::diag::SovdOperationExecution::GetStatus
[SWS_DM_02202]	Definition of API function ara::diag::SovdOperationExecution::Terminate
[SWS_DM_02203]	Definition of API class ara::diag::GenericDiagnosticSovdOperation
[SWS_DM_02204]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::GenericDiagnosticSovdOperation
[SWS_DM_02205]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::~~GenericDiagnosticSovdOperation
[SWS_DM_02206]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::GenericDiagnosticSovdOperation
[SWS_DM_02207]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::operator=
[SWS_DM_02208]	Definition of API class ara::diag::GenericDiagnosticSovdOperation::Start SynchronousExecutionResponse
[SWS_DM_02209]	Definition of API class ara::diag::GenericDiagnosticSovdOperation::Start AsynchronousExecutionResponse





Number	Heading
[SWS_DM_02210]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::Start Execution
[SWS_DM_02211]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::Offer
[SWS_DM_02212]	Definition of API function ara::diag::GenericDiagnosticSovdOperation::Stop Offer
[SWS_DM_02213]	Definition of API enum ara::diag::DiagnosticSovdContentElementDataType
[SWS_DM_02214]	Definition of API enum ara::diag::DiagnosticSovdPrimitiveContentValueType
[SWS_DM_02215]	Definition of API class ara::diag::DiagnosticSovdArrayContentElementData Iterator
[SWS_DM_02216]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataIterator::operator*
[SWS_DM_02217]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataIterator::operator->
[SWS_DM_02218]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataIterator::operator==
[SWS_DM_02219]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataIterator::operator!=
[SWS_DM_02220]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataIterator::operator++
[SWS_DM_02221]	Definition of API class ara::diag::DiagnosticSovdArrayContentElementData ConstIterator
[SWS_DM_02222]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstIterator::operator*
[SWS_DM_02223]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstIterator::operator->
[SWS_DM_02224]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstIterator::operator==
[SWS_DM_02225]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstIterator::operator!=
[SWS_DM_02226]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstIterator::operator++
[SWS_DM_02227]	Definition of API class ara::diag::DiagnosticSovdArrayContentElementData View
[SWS_DM_02228]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::HasError
[SWS_DM_02229]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::SetError
[SWS_DM_02230]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Size
[SWS_DM_02231]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Capacity
[SWS_DM_02232]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Empty
[SWS_DM_02233]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Clear





Number	Heading
[SWS_DM_02234]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::operator[]
[SWS_DM_02235]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::operator[]
[SWS_DM_02236]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Begin
[SWS_DM_02237]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::End
[SWS_DM_02238]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::CBegin
[SWS_DM_02239]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::CEnd
[SWS_DM_02240]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Reserve
[SWS_DM_02241]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02242]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02243]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02244]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02245]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02246]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::PushBack
[SWS_DM_02247]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataView::Erase
[SWS_DM_02248]	Definition of API class ara::diag::DiagnosticSovdArrayContentElementData ConstView
[SWS_DM_02249]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::HasError
[SWS_DM_02250]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::Size
[SWS_DM_02251]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::Capacity
[SWS_DM_02252]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::Empty
[SWS_DM_02253]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::operator[]
[SWS_DM_02254]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::Begin
[SWS_DM_02255]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::End
[SWS_DM_02256]	Definition of API function ara::diag::DiagnosticSovdArrayContentElement DataConstView::CBegin





Number	Heading
[SWS_DM_02257]	Definition of API function ara::diag::DiagnosticSovdArrayContentElementDataConstView::CEnd
[SWS_DM_02258]	Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataIterator
[SWS_DM_02259]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator*
[SWS_DM_02260]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator->
[SWS_DM_02261]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator==
[SWS_DM_02262]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator!=
[SWS_DM_02263]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataIterator::operator++
[SWS_DM_02264]	Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataConstIterator
[SWS_DM_02265]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator*
[SWS_DM_02266]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator->
[SWS_DM_02267]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator==
[SWS_DM_02268]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator!=
[SWS_DM_02269]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstIterator::operator++
[SWS_DM_02270]	Definition of API class ara::diag::DiagnosticSovdRecordContentElementDataView
[SWS_DM_02271]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::HasError
[SWS_DM_02272]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::SetError
[SWS_DM_02273]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Size
[SWS_DM_02274]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Capacity
[SWS_DM_02275]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Empty
[SWS_DM_02276]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::Clear
[SWS_DM_02277]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::operator[]
[SWS_DM_02278]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataView::operator[]





Number	Heading
[SWS_DM_02279]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Begin
[SWS_DM_02280]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::End
[SWS_DM_02281]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::CBegin
[SWS_DM_02282]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::CEnd
[SWS_DM_02283]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Reserve
[SWS_DM_02284]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::HasContentElement
[SWS_DM_02285]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Find
[SWS_DM_02286]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Find
[SWS_DM_02287]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Remove
[SWS_DM_02288]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02289]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02290]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02291]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02292]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02293]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Add
[SWS_DM_02294]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataView::Erase
[SWS_DM_02295]	Definition of API class ara::diag::DiagnosticSovdRecordContentElementData ConstView
[SWS_DM_02296]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::HasError
[SWS_DM_02297]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::Size
[SWS_DM_02298]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::Capacity
[SWS_DM_02299]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::Empty
[SWS_DM_02300]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::operator[]
[SWS_DM_02301]	Definition of API function ara::diag::DiagnosticSovdRecordContentElement DataConstView::Begin





Number	Heading
[SWS_DM_02302]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::End
[SWS_DM_02303]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::CBegin
[SWS_DM_02304]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::CEnd
[SWS_DM_02305]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::HasContentElement
[SWS_DM_02306]	Definition of API function ara::diag::DiagnosticSovdRecordContentElementDataConstView::Find
[SWS_DM_02307]	Definition of API class ara::diag::DiagnosticSovdContentElementData
[SWS_DM_02308]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentSignedInt
[SWS_DM_02309]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentUnsignedInt
[SWS_DM_02310]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentDouble
[SWS_DM_02311]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentBool
[SWS_DM_02312]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreatePrimitiveContentString
[SWS_DM_02313]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateArrayContentElement
[SWS_DM_02314]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateRecordContentElement
[SWS_DM_02315]	Definition of API function ara::diag::DiagnosticSovdContentElementData::CreateContentElement
[SWS_DM_02316]	Definition of API function ara::diag::DiagnosticSovdContentElementData::~DiagnosticSovdContentElementData
[SWS_DM_02317]	Definition of API function ara::diag::DiagnosticSovdContentElementData::HasError
[SWS_DM_02318]	Definition of API function ara::diag::DiagnosticSovdContentElementData::SetError
[SWS_DM_02319]	Definition of API function ara::diag::DiagnosticSovdContentElementData::GetType
[SWS_DM_02320]	Definition of API function ara::diag::DiagnosticSovdContentElementData::GetPrimitiveContentValueType
[SWS_DM_02321]	Definition of API function ara::diag::DiagnosticSovdContentElementData::SetSignedInt
[SWS_DM_02322]	Definition of API function ara::diag::DiagnosticSovdContentElementData::SetUnsignedInt
[SWS_DM_02323]	Definition of API function ara::diag::DiagnosticSovdContentElementData::SetDouble
[SWS_DM_02324]	Definition of API function ara::diag::DiagnosticSovdContentElementData::SetBool





Number	Heading
[SWS_DM_02325]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::SetString
[SWS_DM_02326]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetSignedInt
[SWS_DM_02327]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetUnsignedInt
[SWS_DM_02328]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetDouble
[SWS_DM_02329]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetBool
[SWS_DM_02330]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetString
[SWS_DM_02331]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetArrayView
[SWS_DM_02332]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetRecordView
[SWS_DM_02333]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetArrayView
[SWS_DM_02334]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::GetRecordView
[SWS_DM_02335]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Size
[SWS_DM_02336]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Capacity
[SWS_DM_02337]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Empty
[SWS_DM_02338]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Clear
[SWS_DM_02339]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::operator[]
[SWS_DM_02340]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::operator[]
[SWS_DM_02341]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Begin
[SWS_DM_02342]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::End
[SWS_DM_02343]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::CBegin
[SWS_DM_02344]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::CEnd
[SWS_DM_02345]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Reserve
[SWS_DM_02346]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack





Number	Heading
[SWS_DM_02347]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack
[SWS_DM_02348]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack
[SWS_DM_02349]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack
[SWS_DM_02350]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack
[SWS_DM_02351]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::PushBack
[SWS_DM_02352]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Erase
[SWS_DM_02353]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordCount
[SWS_DM_02354]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordCapacity
[SWS_DM_02355]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordEmpty
[SWS_DM_02356]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordRemoveAll
[SWS_DM_02357]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::operator[]
[SWS_DM_02358]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::operator[]
[SWS_DM_02359]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordBegin
[SWS_DM_02360]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordEnd
[SWS_DM_02361]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordCBegin
[SWS_DM_02362]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordCEnd
[SWS_DM_02363]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::RecordReserve
[SWS_DM_02364]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::HasContentElement
[SWS_DM_02365]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Find
[SWS_DM_02366]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Find
[SWS_DM_02367]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Remove
[SWS_DM_02368]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Add
[SWS_DM_02369]	Definition of API function ara::diag::DiagnosticSovdContentElement Data::Add





Number	Heading
[SWS_DM_02370]	Definition of API function ara::diag::DiagnosticSovdContentElementData::Add
[SWS_DM_02371]	Definition of API function ara::diag::DiagnosticSovdContentElementData::Add
[SWS_DM_02372]	Definition of API function ara::diag::DiagnosticSovdContentElementData::Add
[SWS_DM_02373]	Definition of API function ara::diag::DiagnosticSovdContentElementData::Add
[SWS_DM_02374]	Definition of API function ara::diag::DiagnosticSovdContentElementData::Erase
[SWS_DM_02375]	Definition of API class ara::diag::DiagnosticSovdContentElementDataWithKey
[SWS_DM_02376]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::~DiagnosticSovdContentElementDataWithKey
[SWS_DM_02377]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::GetName
[SWS_DM_02378]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::SetName
[SWS_DM_02379]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::GetContent
[SWS_DM_02380]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::GetContent
[SWS_DM_02381]	Definition of API function ara::diag::DiagnosticSovdContentElementDataWithKey::SetContent
[SWS_DM_02382]	Definition of API enum ara::diag::ProximityChallengeResultType
[SWS_DM_02383]	Definition of API enum ara::diag::OperationExecutionCapability
[SWS_DM_02384]	Definition of API function ara::diag::SovdOperationExecution::ExecuteCapability
[SWS_DM_02385]	Definition of API class ara::diag::DiagnosticSovdContentData
[SWS_DM_02386]	Definition of API variable ara::diag::DiagnosticSovdContentData::elements
[SWS_DM_02387]	Definition of API type ara::diag::ProximityChallengeValidationResultType

Table E.35: Added Specification Items in R25-11

E.11.2 Changed Specification Items in R25-11

Number	Heading
[SWS_DM_00039]	Resetting the internal debounce counter upon restarting an operation cycle
[SWS_DM_00154]	Number of extended data for a DTC
[SWS_DM_00224]	Indicator healing
[SWS_DM_00303]	Definition of API type apex::diag::uds_transport::UdsMessagePtr





Number	Heading
[SWS_DM_00304]	Definition of API type apex::diag::uds_transport::UdsMessageConstPtr
[SWS_DM_00306]	Definition of API class apex::diag::uds_transport::UdsTransportProtocolMgr
[SWS_DM_00309]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage
[SWS_DM_00310]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure
[SWS_DM_00311]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage
[SWS_DM_00312]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation
[SWS_DM_00313]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished
[SWS_DM_00314]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped
[SWS_DM_00319]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::Initialize
[SWS_DM_00322]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::Start
[SWS_DM_00323]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::Stop
[SWS_DM_00325]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::GetHandlerID
[SWS_DM_00326]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment
[SWS_DM_00327]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::Transmit
[SWS_DM_00475]	Support of DoIP based on ISO 13400-2
[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_00552]	Definition of API function ara::diag::GenericRoutine::GenericRoutine
[SWS_DM_00557]	Definition of API function ara::diag::GenericRoutine::Offer
[SWS_DM_00558]	Definition of API function ara::diag::GenericRoutine::StopOffer
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00579]	Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::{<data-identifier-name>} ReadOutput
[SWS_DM_00580]	Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name-x>}::{<data-element-name>} ReadOutput





Number	Heading
[SWS_DM_00581]	Definition of API class { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::{<routine-interface-name-6> }StartOutput
[SWS_DM_00582]	Definition of API class { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::{<routine-interface-name-9> }StopOutput
[SWS_DM_00583]	Definition of API class { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::{<routine-interface-name-11> }RequestResultOutput
[SWS_DM_00585]	Definition of API function {<namespace-list-data-identifier>::{ <data-identifier-interface-name-x>>::{ <data-identifier-interface-name-1>
[SWS_DM_00586]	Definition of API function {<namespace-list-data-identifier>::{ <data-identifier-interface-name-x>::~{ <data-identifier-interface-name-6>
[SWS_DM_00587]	Definition of API function {<namespace-list-data-element>::{ <data-element-interface-name-x>>::{ <data-element-interface-name-1>
[SWS_DM_00588]	Definition of API function {<namespace-list-data-element>::{ <data-element-interface-name-x>::~{ <data-element-interface-name-6>
[SWS_DM_00589]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::{<routine-interface-name-1>
[SWS_DM_00590]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>::~{<routine-interface-name-6> }
[SWS_DM_00591]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::Start
[SWS_DM_00592]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::Stop
[SWS_DM_00593]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::RequestResult
[SWS_DM_00594]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::Offer
[SWS_DM_00595]	Definition of API function { <routine-interface-hierarchical-namespace-list>::{ <routine-interface-name-x>>::StopOffer





Number	Heading
[SWS_DM_00596]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::Read
[SWS_DM_00597]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::Offer
[SWS_DM_00598]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::Write
[SWS_DM_00599]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::Offer
[SWS_DM_00600]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::StopOffer
[SWS_DM_00601]	Definition of API class {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}
[SWS_DM_00603]	Definition of API class {<namespace-list-data-element>}::{<data-element-interface-name-x>}
[SWS_DM_00604]	Definition of API class {<routine-interface-hierarchical-namespace-list>}::{<routine-interface-name-x>}
[SWS_DM_00610]	Definition of API function ara::diag::CancellationHandler::CancellationHandler
[SWS_DM_00612]	Definition of API function ara::diag::CancellationHandler::operator=
[SWS_DM_00616]	Definition of API function ara::diag::GenericUDSService::GenericUDSService
[SWS_DM_00617]	Definition of API function {<namespace-list-data-element>}::{<data-element-interface-name-x>}::StopOffer
[SWS_DM_00634]	Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier
[SWS_DM_00640]	Definition of API function {<namespace-list-data-identifier>}::{<data-identifier-interface-name-x>}::Read
[SWS_DM_00647]	Definition of API function ara::diag::Event::Event
[SWS_DM_00650]	Definition of API function ara::diag::Event::SetEventStatusChangedNotifier
[SWS_DM_00660]	Definition of API class ara::diag::DTCInformation::SnapshotDataIdentifierType
[SWS_DM_00661]	Definition of API class ara::diag::DTCInformation::SnapshotDataRecordType
[SWS_DM_00662]	Definition of API class ara::diag::DTCInformation::SnapshotRecordUpdatedType
[SWS_DM_00664]	Definition of API function ara::diag::DTCInformation::DTCInformation
[SWS_DM_00667]	Definition of API function ara::diag::DTCInformation::SetDTCStatusChangedNotifier
[SWS_DM_00668]	Definition of API function ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier
[SWS_DM_00670]	Definition of API function ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier
[SWS_DM_00673]	Definition of API function ara::diag::DTCInformation::SetControlDtcStatusNotifier





Number	Heading
[SWS_DM_00674]	Definition of API function ara::diag::DTCInformation::EnableControlDtc
[SWS_DM_00695]	Definition of API function ara::diag::Conversation::SetActivityNotifier
[SWS_DM_00697]	Definition of API function ara::diag::Conversation::SetDiagnosticSessionNotifier
[SWS_DM_00699]	Definition of API function ara::diag::Conversation::SetSecurityLevelNotifier
[SWS_DM_00701]	Definition of API function ara::diag::Conversation::ResetToDefaultSession
[SWS_DM_00712]	Definition of API function ara::diag::Condition::Condition
[SWS_DM_00715]	Definition of API function ara::diag::Condition::SetCondition
[SWS_DM_00722]	Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification
[SWS_DM_00724]	Definition of API function ara::diag::DoIPGroupIdentification::GetGidStatus
[SWS_DM_00732]	Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode
[SWS_DM_00734]	Definition of API function ara::diag::DoIPPowerMode::GetDoIPPowerMode
[SWS_DM_00742]	Definition of API function ara::diag::Indicator::Indicator
[SWS_DM_00745]	Definition of API function ara::diag::Indicator::SetNotifier
[SWS_DM_00752]	Definition of API function ara::diag::OperationCycle::OperationCycle
[SWS_DM_00755]	Definition of API function ara::diag::OperationCycle::SetNotifier
[SWS_DM_00762]	Definition of API function ara::diag::SecurityAccess::SecurityAccess
[SWS_DM_00772]	Definition of API function ara::diag::ServiceValidation::ServiceValidation
[SWS_DM_00775]	Definition of API function ara::diag::ServiceValidation::Confirmation
[SWS_DM_00784]	Definition of API class ara::diag::DownloadService
[SWS_DM_00787]	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_00794]	Definition of API class ara::diag::UploadService
[SWS_DM_00797]	Definition of API function ara::diag::UploadService::UploadService
[SWS_DM_00804]	Definition of API class ara::diag::CommunicationControl
[SWS_DM_00806]	Definition of API function ara::diag::CommunicationControl::CommunicationControl
[SWS_DM_00815]	Triggering vehicle announcement messages
[SWS_DM_00816]	Notification of activation line status change on activation line controlled network interfaces
[SWS_DM_00821]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::GetDoIPTriggerVehicleAnnouncement
[SWS_DM_00822]	Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement
[SWS_DM_00831]	Definition of API function ara::diag::DoIPActivationLine::DoIPActivationLine
[SWS_DM_00833]	Definition of API function ara::diag::DoIPActivationLine::GetNetworkInterfaceId





Number	Heading
[SWS_DM_00834]	Definition of API function ara::diag::DoIPActivationLine::UpdateActivationLineState
[SWS_DM_00835]	Definition of API function ara::diag::DoIPActivationLine::GetActivationLineState
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00856]	Initial values for Diagnostic Conversation
[SWS_DM_00895]	Triggering for extended data record storage and updates
[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
[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_00974]	Definition of API function ara::diag::MetaInfo::MetaInfo
[SWS_DM_01009]	Definition of API class ara::diag::EcuResetRequest
[SWS_DM_01010]	Definition of API function ara::diag::EcuResetRequest::EcuResetRequest
[SWS_DM_01062]	Transmission of all PDIDs on the periodic connection
[SWS_DM_01065]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages
[SWS_DM_01066]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength
[SWS_DM_01067]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit
[SWS_DM_01068]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolHandler::GetPeriodicHandler
[SWS_DM_01069]	Definition of API function apex::diag::uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation
[SWS_DM_01102]	Definition of API function ara::diag::OperationCycle::RestartOperationCycle
[SWS_DM_01124]	Definition of API function ara::diag::Authentication::Authentication





Number	Heading
[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_01144]	Definition of API function ara::diag::ClientAuthentication::SetNotifier
[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_01159]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::DiagnosticServiceDynamicAccessList
[SWS_DM_01160]	Definition of API function ara::diag::DiagnosticServiceDynamicAccess List::DiagnosticServiceDynamicAccessList
[SWS_DM_01170]	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_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_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_01325]	Definition of API function ara::diag::FileTransferService::FileTransferService
[SWS_DM_01328]	Definition of API function ara::diag::FileTransferService::FileTransferService
[SWS_DM_01331]	Definition of API function ara::diag::FileTransferService::RequestReadFile
[SWS_DM_01364]	Definition of API function ara::diag::DoIPEntityIdentification::DoIPEntity Identification
[SWS_DM_01366]	Definition of API function ara::diag::DoIPEntityIdentification::GetEntityId
[SWS_DM_01390]	SOVD operations request and response parameters
[SWS_DM_01391]	SOVD method Terminate execution of an Operation – proximity_response
[SWS_DM_01392]	Support of SOVD method Terminate execution of an Operation
[SWS_DM_01393]	Support of SOVD method Read the Status of an Operation Execution
[SWS_DM_01394]	Support of SOVD method Query for executions of an operation





Number	Heading
[SWS_DM_01395]	Validation of SOVD operation <code>proximity_response</code>
[SWS_DM_01396]	Support of SOVD method <code>Start Execution of an Operation</code>
[SWS_DM_01398]	Providing the SOVD operation <code>proximity_challenge</code>
[SWS_DM_01399]	SOVD operation attribute <code>asynchronous_execution</code>
[SWS_DM_01400]	SOVD operation attribute <code>proximity_proof_required</code>
[SWS_DM_01401]	SOVD operation attribute <code>name</code>
[SWS_DM_01402]	SOVD operation attribute <code>id</code>
[SWS_DM_01403]	Support of SOVD method <code>Retrieve details of an operation</code>
[SWS_DM_01404]	Support of SOVD method <code>Query for operations from an Entity</code>
[SWS_DM_01405]	SOVD operations
[SWS_DM_01406]	Support of SOVD method <code>Delete Single Fault of an Entity</code>
[SWS_DM_01409]	Support of SOVD method <code>Delete All Faults of an Entity</code>
[SWS_DM_01431]	Support of SOVD method <code>Write a data resource from an Entity</code>
[SWS_DM_01432]	SOVD method <code>Read a data resource from an Entity – data query</code>
[SWS_DM_01433]	Support of SOVD method <code>Read a data resource from an Entity</code>
[SWS_DM_01442]	Support of SOVD method <code>Query for data from an Entity</code>
[SWS_DM_01446]	Support of SOVD method <code>Retrieve Groups Supported by a Data Resource Collection</code>
[SWS_DM_01448]	Retrieve resource <code>Data categories</code>
[SWS_DM_01449]	Support of SOVD method <code>Retrieve Categories Supported by a Data Resource Collection</code>
[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 <code>baseTypeEncoding</code> IEEE754
[SWS_DM_01461]	SOVD data representation of <code>BITFIELD_TEXTTABLE</code>
[SWS_DM_01462]	SOVD data representation of <code>TAB_NOINTP</code>
[SWS_DM_01463]	SOVD data representation of atomic scaled numeric data with <code>TEXTTABLE</code>
[SWS_DM_01464]	SOVD data representation of <code>TEXTTABLE</code>
[SWS_DM_01465]	SOVD data representation of atomic numeric data
[SWS_DM_01468]	Check of Environmental Conditions before executing SOVD methods
[SWS_DM_01480]	Definition of API function <code>ara::diag::SovdProximityChallenge::SovdProximityChallenge</code>
[SWS_DM_01481]	Definition of API class <code>ara::diag::SovdProximityChallenge</code>
[SWS_DM_01483]	Definition of API function <code>ara::diag::SovdAuthorization::ValidateAuthorization</code>
[SWS_DM_01484]	Definition of API function <code>ara::diag::SovdAuthorization::GetTokenUrl</code>
[SWS_DM_01485]	Definition of API function <code>ara::diag::SovdAuthorization::GetAuthorizationUrl</code>
[SWS_DM_01489]	Definition of API function <code>ara::diag::SovdAuthorization::SovdAuthorization</code>





Number	Heading
[SWS_DM_01493]	Definition of API function ara::diag::SovdProximityChallenge::Validate Response
[SWS_DM_01494]	Definition of API function ara::diag::SovdProximityChallenge::GetChallenge
[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_01502]	Definition of API function ara::diag::DataTransferReadSharedData Handler::operator=
[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_01510]	Definition of API function ara::diag::DataTransferReadByPullHandler::~Data TransferReadByPullHandler
[SWS_DM_01511]	Definition of API function ara::diag::DataTransferReadByPull Handler::operator=
[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_01519]	Definition of API function ara::diag::DataTransferReadByPush Handler::~DataTransferReadByPushHandler
[SWS_DM_01520]	Definition of API function ara::diag::DataTransferReadByPush Handler::operator=
[SWS_DM_01524]	Definition of API function apext::diag::uds_transport::UdsTransportProtocol Mgr::~UdsTransportProtocolMgr
[SWS_DM_01532]	Definition of API function ara::diag::ReleaseHandler::ReleaseHandler
[SWS_DM_01540]	Definition of API function ara::diag::DataTransferWriteHandler::Data Transfer WriteHandler
[SWS_DM_01542]	Definition of API function ara::diag::DataTransferWriteHandler::Data Transfer WriteHandler
[SWS_DM_01543]	Definition of API function ara::diag::DataTransferWriteHandler::operator=
[SWS_DM_01550]	Definition of API function ara::diag::DataTransferReadSession::~Data TransferReadSession
[SWS_DM_01551]	Definition of API function ara::diag::DataTransferReadSession::Data Transfer ReadSession
[SWS_DM_01552]	Definition of API function ara::diag::DataTransferReadSession::operator=
[SWS_DM_01556]	Definition of API function ara::diag::DataTransferWriteSession::~Data TransferWriteSession
[SWS_DM_01557]	Definition of API function ara::diag::DataTransferWriteSession::Data Transfer WriteSession





Number	Heading
[SWS_DM_01558]	Definition of API function ara::diag::DataTransferWriteSession::operator=
[SWS_DM_01561]	Response Body for SOVD fault <code>environment_data</code> table
[SWS_DM_01565]	Supported <code>internal</code> data for UDS ReadDataByIdentifier
[SWS_DM_01568]	<code>supportedDolpMessageTypes</code>
[SWS_DM_01582]	Healing counter increment
[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_01702]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_01705]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_01708]	Definition of API function ara::diag::MultipleEvent::SetEventStatusChanged Notifier
[SWS_DM_01727]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_01751]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01752]	Definition of API function ara::diag::EventStatusByte::EventStatusByte
[SWS_DM_01823]	Definition of API function ara::diag::SovdServiceValidation::SovdService Validation
[SWS_DM_01827]	Definition of API function ara::diag::SovdServiceValidation::Validate
[SWS_DM_01847]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01849]	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_01864]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[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_01873]	Definition of API function ara::diag::SovdBulkData::RequestBulkData Download
[SWS_DM_01910]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[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





Number	Heading
[SWS_DM_01921]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackageStatus
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration
[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_01962]	Definition of API function ara::diag::TransmitCertificate::TransmitCertificate
[SWS_DM_01967]	Definition of API function ara::diag::TransmitCertificate::~TransmitCertificate
[SWS_DM_01972]	Definition of API function ara::diag::Monitor::Monitor
[SWS_DM_01973]	Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor
[SWS_DM_02004]	UDS request priority handling
[SWS_DM_02006]	Providing EID using manually configured value
[SWS_DM_02010]	Vehicle Announcement/Identification Response Message with "VIN/GID sync status" byte
[SWS_DM_02014]	Security events for Diagnostic Management
[SWS_DM_02080]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsFailedAndTested
[SWS_DM_02081]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsPassedAndTested
[SWS_DM_02082]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsSet
[SWS_DM_02083]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::IsNotSet
[SWS_DM_02085]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType
[SWS_DM_02086]	Definition of API function ara::diag::DTCInformation::UdsDtcStatusByteType::UdsDtcStatusByteType
[SWS_DM_02089]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolPeriodicHandler::~UdsTransportProtocolPeriodicHandler
[SWS_DM_09010]	Definition of API function apext::diag::uds_transport::UdsMessage::~UdsMessage
[SWS_DM_09012]	Definition of API function apext::diag::uds_transport::UdsMessage::UdsMessage
[SWS_DM_09015]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler
[SWS_DM_09016]	Definition of API function apext::diag::uds_transport::UdsTransportProtocolHandler::~UdsTransportProtocolHandler

Table E.36: Changed Specification Items in R25-11

E.11.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_DM_00127]	Availability of diagnostic service processors
[SWS_DM_00651]	Definition of API function <code>ara::diag::Event::GetLatchedWIRStatus</code>
[SWS_DM_00652]	Definition of API function <code>ara::diag::Event::SetLatchedWIRStatus</code>
[SWS_DM_00841]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00844]	Updating DiagnosticConversation Service Instance fields
[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_01413]	SOVD fault attribute <code>symptom</code>
[SWS_DM_01421]	SOVD method <code>Write Configuration as Parameters</code> data processing
[SWS_DM_01422]	SOVD method <code>Write Configuration as Parameters</code>
[SWS_DM_01423]	SOVD method <code>Read Configuration as Parameters</code> data query
[SWS_DM_01424]	SOVD method <code>Read Configuration as Parameters</code>
[SWS_DM_01425]	SOVD configuration attribute <code>version</code> and <code>content_type</code>
[SWS_DM_01426]	SOVD configuration attribute <code>type</code>
[SWS_DM_01427]	SOVD configuration attribute <code>name</code>
[SWS_DM_01428]	SOVD configuration attribute <code>id</code>
[SWS_DM_01430]	SOVD method <code>Write a Data Value to an Entity</code> data processing
[SWS_DM_01434]	SOVD data attribute <code>data</code> of internal structure
[SWS_DM_01435]	SOVD <code>group</code> <code>id</code>
[SWS_DM_01436]	SOVD <code>group</code> attribute <code>category</code>
[SWS_DM_01437]	SOVD <code>group</code> <code>category</code> uniqueness
[SWS_DM_01439]	SOVD data attribute <code>category</code>
[SWS_DM_01440]	SOVD data attribute <code>name</code>
[SWS_DM_01441]	SOVD data attribute <code>id</code>
[SWS_DM_01444]	Data-groups <code>type</code> <code>content</code>
[SWS_DM_01445]	Data-groups <code>type</code>
[SWS_DM_01447]	Data <code>categories</code> <code>type</code>
[SWS_DM_01526]	<code>ara::diag::DoIPActivationLine</code> not yet offered on activation line controlled network interfaces
[SWS_DM_01529]	Behavior on failed <code>ara::diag</code> instantiation
[SWS_DM_02011]	<code>VIN/GID</code> status on successful <code>VIN</code> retrieval
[SWS_DM_02012]	<code>VIN/GID</code> status on unsuccessful <code>VIN</code> and <code>GID</code> synchronization
[SWS_DM_02013]	<code>VIN/GID</code> status on unsuccessful <code>VIN</code> and successful <code>GID</code> synchronization

Table E.37: Deleted Specification Items in R25-11

E.11.4 Added Constraints in R25-11

Number	Heading
[SWS_DM_CONSTR_02143]	Mapping bus type to CPP Implementation data type for scalar types
[SWS_DM_CONSTR_02144]	Mapping bus types to CPP Implementation data type for scalar types

Table E.38: Added Constraints in R25-11

E.11.5 Changed Constraints in R25-11

Number	Heading
[SWS_DM_CONSTR_00960]	No support for DEM_AGINGCTR_UPCNT_FIRST_ACTIVE

Table E.39: Changed Constraints in R25-11

E.11.6 Deleted Constraints in R25-11

none