

<b>Document Title</b>	Specification of Diagnostics
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	723

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Adaptive Platform
<b>Part of Standard Release</b>	R23-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Incorporated Quality Scope Review Findings</li> <li>• SOVD Concept Part 2 implemented</li> <li>• Service 0x29 refinements</li> </ul>
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Incorporated Quality Scope Review Findings</li> <li>• Introduced DTC suppressed feature</li> <li>• Standardize mapping of vendor specific error codes to UDS Error codes</li> <li>• Introduced 0x38 RequestFileTransfer</li> <li>• Introduced SOVD Concept</li> </ul>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Incorporated Quality Scope Review Findings</li> <li>• Introduced UDS service 29</li> <li>• Introduced Event Combination in chapter 7</li> </ul>





2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Incorporated Quality Scope Review Findings</li> <li>• Validated requirements from concept DoIPEExtension</li> <li>• Introduced UDS services 2A &amp; 2C</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Incorporated Quality Scope Review Findings</li> <li>• Partly removed obsolete requirements</li> <li>• Removed obsolete service interfaces</li> <li>• Changed Document Status from Final to published</li> </ul>
2019-03-29	19-03	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Document quality improvement and fixing bugs</li> <li>• Introduced ara::diag interfaces in draft state</li> </ul>
2018-10-31	18-10	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Diagnostic Protocol replaced by Diagnostic Conversations</li> <li>• ResponseOnEvent, CommunicationControl, EcuReset added</li> <li>• Chapter 7 overall rework and updates</li> <li>• Chapter 8 split into chapter 8 (C++ API) and chapter 9 (Service Interfaces)</li> </ul>
2018-03-29	18-03	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Chapter 7.1. Software Cluster added</li> <li>• Chapter 7.2. Diagnostic Service Management, common parts for all services separated</li> <li>• Chapter 7.3. Event Management, several additions and rework</li> <li>• Chapter 8. API specification, complete rework</li> </ul>



△

2017-10-27	17-10	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>● General API rework</li> <li>● TP Plug-in interface</li> <li>● Introduction of SoftwareCluster in APIs</li> <li>● Additional UDS services like SecurityAccess</li> </ul>
2017-03-31	17-03	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>● Initial release</li> </ul>

## 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.

## Contents

1	Introduction and functional overview	35
1.1	Diagnostic interface	35
1.2	AUTOSAR Diagnostic Extract Template (DEXT)	35
1.3	Software Cluster	35
1.3.1	Diagnostic Server	35
1.3.2	Diagnostic Managers external dependencies	38
2	Acronyms and Abbreviations	39
3	Related documentation	44
3.1	Input documents & related standards and norms	44
3.2	Further applicable specification	44
4	Constraints and assumptions	46
4.1	Known Limitations	46
5	Dependencies to other Functional Clusters	49
5.1	Provided Interfaces	49
5.2	Required Interfaces	49
6	Requirements Tracing	52
6.1	Not applicable requirements	71
7	Functional specification	72
7.1	Functional cluster life-cycle	73
7.1.1	Startup	73
7.1.2	Shutdown	73
7.1.3	Daemon crash	74
7.2	UDS Transport Layer	74
7.2.1	Support of proprietary UDS Transport Layer	75
7.2.1.1	Initialization, Starting and Stopping of a proprietary UDS TransportLayer	75
7.2.1.2	UDS message reception on a proprietary UDS TransportLayer	77
7.2.1.3	UDS message transmission on a proprietary UDS TransportLayer	80
7.2.1.4	Channel Notifications	81
7.2.2	DoIP	82
7.2.3	Dispatching of UDS Requests	84
7.3	SOVD Transport Layer	85
7.4	Diagnostic Server	86
7.4.1	Interaction between DM and applications	87
7.4.2	Diagnostic Communication Management	87
7.4.2.1	Diagnostic Conversations	87
7.4.2.1.1	Parallel Client Handling	88

7.4.2.1.2	Life-cycle of a Diagnostic Conversation . . . . .	91
7.4.2.1.3	Diagnostic Conversation Service Interface . . . . .	92
7.4.2.2	Assignment of UDS requests to Diagnostic Conversations . . . . .	93
7.4.2.2.1	Prioritization . . . . .	96
7.4.2.2.2	Replacement of Diagnostic Conversations and initial values . . . . .	97
7.4.2.2.3	Refusal of incoming diagnostic request . . . . .	98
7.4.2.3	Handling Authentication State and DynamicAccessLists . . . . .	98
7.4.2.3.1	ExternalAuthentication . . . . .	98
7.4.2.3.2	ClientAuthentication . . . . .	99
7.4.2.3.3	ClientAuthenticationHandle . . . . .	101
7.4.2.3.4	DynamicAccessList Creation and Update . . . . .	102
7.4.2.4	Diagnostic Service Authentication checks . . . . .	103
7.4.2.5	UDS request Validation/Verification . . . . .	105
7.4.2.5.1	UDS request format checks . . . . .	105
7.4.2.5.2	Supported service checks . . . . .	106
7.4.2.5.3	Session and Security Checks . . . . .	106
7.4.2.5.4	Manufacturer and Supplier Permission Checks and Confirmation . . . . .	108
7.4.2.5.5	Condition checks . . . . .	109
7.4.2.6	UDS response handling . . . . .	110
7.4.2.7	Keep track of active non-default sessions . . . . .	111
7.4.2.8	UDS service processing . . . . .	111
7.4.2.8.1	Supported UDS Services . . . . .	112
7.4.2.8.2	Common service processing items . . . . .	112
7.4.2.8.3	UDS service serialization . . . . .	113
7.4.2.8.4	Service 0x10 – DiagnosticSessionControl . . . . .	114
7.4.2.8.5	Service 0x11 – ECUReset . . . . .	115
7.4.2.8.6	Service 0x14 – ClearDiagnosticInformation . . . . .	117
7.4.2.8.6.1	Clearing user-defined fault memory . . . . .	119
7.4.2.8.7	Service 0x19 – ReadDTCInformation . . . . .	120
7.4.2.8.7.1	SF 0x01 – reportNumberOfDTCByStatusMask . . . . .	121
7.4.2.8.7.2	SF 0x02 – reportDTCByStatusMask . . . . .	121
7.4.2.8.7.3	SF 0x03 – reportDTCSnapshotIdentification . . . . .	121
7.4.2.8.7.4	SF 0x04 – reportDTCSnapshotRecordByDTCNumber . . . . .	122
7.4.2.8.7.5	SF 0x06 – reportDTCExtDataRecordByDTCNumber . . . . .	122
7.4.2.8.7.6	SF 0x07 – reportNumberOfDTCBySeverityMaskRecord . . . . .	122
7.4.2.8.7.7	SF 0x0A – reportSupportedDTC . . . . .	123

7.4.2.8.7.8	SF 0x14 – reportDTCFaultDetection-Counter	123
7.4.2.8.7.9	SF 0x17 – reportUserDefMemory-DTCByStatusMask	123
7.4.2.8.7.10	SF 0x18 – reportUserDefMemoryDTC-SnapshotRecordByDTCNumber	123
7.4.2.8.7.11	SF 0x19 – reportUserDefMemory-DTCExtDataRecordByDTCNumber	124
7.4.2.8.8	Service 0x22 – ReadDataByIdentifier	124
7.4.2.8.9	Service 0x27 – SecurityAccess	125
7.4.2.8.10	Service 0x28 – CommunicationControl	129
7.4.2.8.11	Service 0x29 – Authentication	130
7.4.2.8.11.1	VerifyCertificateUnidirectional	131
7.4.2.8.11.2	VerifyCertificateBidirectional	132
7.4.2.8.11.3	ProofOfOwnership	133
7.4.2.8.11.4	DeAuthenticate	134
7.4.2.8.11.5	AuthenticationConfiguration	134
7.4.2.8.11.6	TransmitCertificate	135
7.4.2.8.12	Service 0x2A – ReadDataByPeriodicIdentifier	135
7.4.2.8.13	Service 0x2C – DynamicallyDefineDataIdentifier	139
7.4.2.8.14	Service 0x2E – WriteDataByIdentifier	141
7.4.2.8.15	Service 0x31 – RoutineControl	142
7.4.2.8.16	Service 0x34 – RequestDownload	144
7.4.2.8.17	Service 0x35 – RequestUpload	144
7.4.2.8.18	Service 0x36 – TransferData	145
7.4.2.8.19	Service 0x37 – RequestTransferExit	145
7.4.2.8.20	Service 0x38 – RequestFileTransfer	146
7.4.2.8.21	Service 0x3E – TesterPresent	148
7.4.2.8.22	Service 0x85 – ControlDTCSetting	148
7.4.2.8.23	Service 0x86 – ResponseOnEvent	150
7.4.2.8.24	Custom Diagnostic Services	153
7.4.2.9	Cancellation of a Diagnostic Conversation	153
7.4.3	Diagnostic SOVD Management	155
7.4.3.1	SOVD Conversations and UDS Interplay	155
7.4.3.2	SOVD Request Validation and Verification	155
7.4.3.2.1	SOVD Authorization	155
7.4.3.2.2	SOVD Lock	156
7.4.3.2.3	Validation of Environmental Conditions in SOVD	157
7.4.3.2.4	Service Validation of SOVD Requests	157
7.4.3.3	SOVD Data Conversion	158
7.4.3.4	Standardized APIs	160
7.4.3.4.1	Docs	160
7.4.3.4.2	Version-Info	160
7.4.3.4.3	Data-categories	161
7.4.3.4.4	Data-groups	161

	7.4.3.4.5	Locks	161
	7.4.3.4.6	Logging	162
7.4.3.5		Configurable APIs	162
	7.4.3.5.1	Data	163
	7.4.3.5.2	Configuration	164
	7.4.3.5.2.1	Derived Configurations	165
	7.4.3.5.2.2	Explicit Configurations	166
	7.4.3.5.3	Data-list	166
	7.4.3.5.4	Faults	167
	7.4.3.5.5	Operations	170
	7.4.3.5.6	Modes	172
	7.4.3.5.7	CommunicationControl	172
	7.4.3.5.8	ControlDTCSetting	173
	7.4.3.5.9	Bulk Data	174
	7.4.3.5.10	Software Update	175
7.4.4		Diagnostic Event Management	176
	7.4.4.1	Caching of application call for event processing	176
	7.4.4.2	Diagnostic Events	177
	7.4.4.2.1	Event Definition	177
	7.4.4.2.2	Monitors	179
	7.4.4.2.3	Event Combination	183
	7.4.4.2.3.1	Combination On Storage	185
	7.4.4.2.3.2	Combination On Retrieval	186
	7.4.4.2.4	EnableConditions	187
	7.4.4.2.5	Debouncing	188
	7.4.4.2.5.1	Counter-based debouncing	190
	7.4.4.2.5.2	Time-based debouncing	193
	7.4.4.2.5.3	Monitor-internal debouncing	196
	7.4.4.2.5.4	Debounce algorithm reset	197
	7.4.4.2.5.5	Dependencies to enable conditions	197
	7.4.4.2.5.6	Dependencies to UDS service 0x85 ControlDTCSettings	198
	7.4.4.2.6	Event Status processing	198
	7.4.4.2.7	Event status change notifications	199
	7.4.4.2.8	Event occurrence	200
7.4.4.3		Condition Mangement	200
7.4.4.4		Operation Cycles Management	201
7.4.4.5		Event memory	202
	7.4.4.5.1	DTC Introduction	203
	7.4.4.5.1.1	Format	203
	7.4.4.5.1.2	Groups	203
	7.4.4.5.1.3	Priority	204
	7.4.4.5.2	UDS DTC Status	205
	7.4.4.5.2.1	Status processing	205
	7.4.4.5.2.2	UDS DTC Status change notifications	206
	7.4.4.5.2.3	Suppress DTC reporting	206



7.4.4.5.2.4	Indicators	208
7.4.4.5.2.5	User	controlled
	WarningIndicatorRequest-bit	210
7.4.4.5.3	Destination	211
7.4.4.5.4	DTC related data	211
7.4.4.5.4.1	Triggering for data storage	211
7.4.4.5.4.2	Storage of snapshot record data	212
7.4.4.5.4.3	Storage of extended data	213
7.4.4.5.4.4	Internal statistical data elements in EDRs	214
7.4.4.5.5	Clearing DTCs	218
7.4.4.5.5.1	Locking of the DTC clearing process by a client	218
7.4.4.5.5.2	ClearConditions	219
7.4.4.5.5.3	DTC clearing triggered by application	219
7.4.4.5.6	Aging	220
7.4.4.5.7	NumberOfStoredEntries	222
7.4.4.5.8	Active / Passive Status of Events	222
7.4.4.5.9	Event memory overflow	222
7.4.4.5.10	Event memory entry displacement	224
7.4.4.5.11	Reporting order of event memory entries	227
7.4.5	Required Configuration	228
7.4.6	Diagnostic Data Management	228
7.4.6.1	Internal and External Diagnostic Data Elements	229
7.4.6.2	Reading and Writing Diagnostic Data Identifier	231
7.4.6.2.1	Supported Diagnostic Mappings	231
7.4.6.2.2	Reading Diagnostic Data Identifier	232
7.4.6.2.3	Writing Diagnostic Data Identifier	233
7.5	Security Events	234
8	API specification	236
8.1	C++ language binding <sub component>	236
8.2	API Common Data Types	236
8.2.1	C++ Diagnostic Error Types	236
8.3	API Reference	258
8.3.1	C++ Diagnostic API Interfaces	258
8.3.1.1	Introduction	258
8.3.1.2	MetaInfo class	259
8.3.1.2.1	diag::MetaInfo class	260
8.3.1.2.2	diag::MetaInfo::MetaInfo constructor	260
8.3.1.2.3	diag::MetaInfo::MetaInfo copy constructor	260
8.3.1.2.4	diag::MetaInfo::MetaInfo move constructor	261
8.3.1.2.5	diag::MetaInfo::MetaInfo copy assignment operator	261
8.3.1.2.6	diag::MetaInfo::MetaInfo move assignment operator	261
8.3.1.2.7	diag::MetaInfo::Context type	262

8.3.1.2.8	diag::MetaInfo::Context kDiagnosticCommunication	262
8.3.1.2.9	diag::MetaInfo::Context kFaultMemory	262
8.3.1.2.10	diag::MetaInfo::Context kDoIP	263
8.3.1.2.11	diag::MetaInfo::Context kSovd	263
8.3.1.2.12	diag::MetaInfo::GetValue function	263
8.3.1.2.13	diag::MetaInfo::GetContext function	264
8.3.1.2.14	diag::MetaInfo::~~MetaInfo destructor	264
8.3.1.3	ReentrancyType class	265
8.3.1.3.1	diag::DataIdentifierReentrancyType class	266
8.3.1.4	Monitor class	267
8.3.1.4.1	diag::Monitor::Monitor constructors	268
8.3.1.4.2	diag::Monitor::ReportMonitorAction function	269
8.3.1.4.3	diag::Monitor::Offer function	269
8.3.1.4.4	diag::Monitor::StopOffer function	270
8.3.1.4.5	diag::Monitor::operator= move assignment operator	270
8.3.1.4.6	diag::Monitor::operator= copy assignment operator	270
8.3.1.4.7	diag::Monitor::Monitor(Monitor&&) move constructor	271
8.3.1.4.8	diag::Monitor::Monitor(Monitor&) copy constructor	271
8.3.1.5	Monitor types class	271
8.3.1.5.1	diag::CounterBased type	271
8.3.1.5.2	diag::TimeBased type	274
8.3.1.5.3	diag::InitMonitorReason enum	275
8.3.1.5.4	diag::MonitorAction enum	275
8.3.1.6	MultipleMonitor class	276
8.3.1.6.1	diag::MonitorHandleType type	276
8.3.1.6.2	diag::MultipleMonitor::MultipleMonitor type	277
8.3.1.6.3	diag::MultipleMonitor::ConfigureMonitor function	277
8.3.1.6.4	diag::MultipleMonitor::ConfigureMonitor function	278
8.3.1.6.5	diag::MultipleMonitor::ConfigureMonitor function	278
8.3.1.6.6	diag::MultipleMonitor::ReportMonitorAction function	279
8.3.1.6.7	diag::MultipleMonitor::Offer function	279
8.3.1.6.8	diag::MultipleMonitor::StopOffer function	280
8.3.1.7	GenericUDSService class	280
8.3.1.7.1	diag::GenericUDSService::OperationOutput type	280
8.3.1.7.2	diag::GenericUDSService::OperationOutput::responseData	281
8.3.1.7.3	diag::GenericUDSService::GenericUDSService function	281

8.3.1.7.4	diag::GenericUDSService::~~GenericUDSService function . . . . .	282
8.3.1.7.5	diag::GenericUDSService::GenericUDSService(GenericUDSServ move constructor . . . . .	282
8.3.1.7.6	diag::GenericUDSService::GenericUDSService(GenericUDSServ copy constructor . . . . .	282
8.3.1.7.7	diag::GenericUDSService::GenericUDSService& operator=(GenericUDSService&&) move op- erator . . . . .	283
8.3.1.7.8	diag::GenericUDSService::GenericUDSService& operator=(GenericUDSService&) copy operator	283
8.3.1.7.9	diag::GenericUDSService::Offer function . . .	283
8.3.1.7.10	diag::GenericUDSService::StopOffer function	284
8.3.1.7.11	diag::GenericUDSService::HandleMessage function . . . . .	284
8.3.1.8	GenericDataIdentifier class . . . . .	287
8.3.1.8.1	diag::GenericDataIdentifier::OperationOutput type . . . . .	288
8.3.1.8.2	diag::GenericDataIdentifier::GenericData Identifier function . . . . .	289
8.3.1.8.3	diag::GenericDataIdentifier::~~GenericData Identifier function . . . . .	289
8.3.1.8.4	diag::GenericDataIdentifier::GenericData Identifier(GenericDataIdentifier&&) move constructor . . . . .	289
8.3.1.8.5	diag::GenericDataIdentifier::GenericData Identifier(GenericDataIdentifier&) copy con- structor . . . . .	290
8.3.1.8.6	diag::GenericDataIdentifier::GenericDataIdentifier& operator=(GenericDataIdentifier&&) move op- erator . . . . .	290
8.3.1.8.7	diag::GenericDataIdentifier::GenericDataIdentifier& operator=(GenericDataIdentifier&&) copy op- erator . . . . .	291
8.3.1.8.8	diag::GenericDataIdentifier::Offer function . .	291
8.3.1.8.9	diag::GenericDataIdentifier::StopOffer function	291
8.3.1.8.10	diag::GenericDataIdentifier::Read function . .	292
8.3.1.8.11	diag::GenericDataIdentifier::Write function . .	295
8.3.1.9	GenericRoutine class . . . . .	298
8.3.1.9.1	diag::GenericRoutine::OperationOutput type .	299
8.3.1.9.2	diag::GenericRoutine::GenericRoutine function	299
8.3.1.9.3	diag::GenericRoutine::GenericRoutine(GenericRoutine&&) move constructor . . . . .	300
8.3.1.9.4	diag::GenericRoutine::GenericRoutine(GenericRoutine&) copy constructor . . . . .	300

8.3.1.9.5	diag::GenericRoutine::GenericRoutine& operator=(GenericRoutine&&) move operator . . .	301
8.3.1.9.6	diag::GenericRoutine::GenericRoutine& operator=(GenericRoutine&) move operator . . . . .	301
8.3.1.9.7	diag::GenericRoutine::~~GenericRoutine function	301
8.3.1.9.8	diag::GenericRoutine::Offer function . . . . .	302
8.3.1.9.9	diag::GenericRoutine::StopOffer function . . .	302
8.3.1.9.10	diag::GenericRoutine::Start function . . . . .	302
8.3.1.9.11	diag::GenericRoutine::Stop function . . . . .	306
8.3.1.9.12	diag::GenericRoutine::RequestResults function	309
8.3.1.10	CancellationHandler class . . . . .	312
8.3.1.10.1	diag::CancellationHandler::CancellationHandler function . . . . .	313
8.3.1.10.2	diag::CancellationHandler::IsCanceled function	314
8.3.1.10.3	diag::CancellationHandler::SetNotifier function	315
8.3.1.11	Event class . . . . .	315
8.3.1.11.1	diag::Event::EventStatusByte Constructor . . .	315
8.3.1.11.2	diag::Event::EventStatusByte Move constructor	316
8.3.1.11.3	diag::Event::EventStatusByte Copy constructor	316
8.3.1.11.4	diag::Event::EventStatusByte::IsFailedAndTested function . . . . .	317
8.3.1.11.5	diag::Event::EventStatusByte::IsNotSet function	317
8.3.1.11.6	diag::Event::EventStatusByte::IsPassedAndTested function . . . . .	317
8.3.1.11.7	diag::Event::EventStatusByte::IsSet function .	318
8.3.1.11.8	diag::Event::EventStatusByte Move assignment operator . . . . .	318
8.3.1.11.9	diag::Event::EventStatusByte Copy assignment operator . . . . .	319
8.3.1.11.10	diag::Event::Event constructor . . . . .	319
8.3.1.11.11	diag::Event::Event(Event&&) move constructor	319
8.3.1.11.12	diag::Event::Event(Event&) copy constructor .	320
8.3.1.11.13	diag::Event::Event& operator=(Event&&) move operator . . . . .	320
8.3.1.11.14	diag::Event::Event& operator=(Event&) copy operator . . . . .	320
8.3.1.11.15	diag::Event::~~Event destructor . . . . .	321
8.3.1.11.16	diag::Event::GetEventStatus function . . . . .	321
8.3.1.11.17	diag::Event::SetEventStatusChangedNotifier function . . . . .	321
8.3.1.11.18	diag::Event::GetLatchedWIRStatus function .	322
8.3.1.11.19	diag::Event::SetLatchedWIRStatus function . .	322
8.3.1.11.20	diag::Event::GetDTCNumber function . . . . .	323
8.3.1.11.21	diag::Event::GetDebouncingStatus function . .	323
8.3.1.11.22	diag::Event::GetTestComplete function . . . .	324
8.3.1.11.23	diag::Event::GetFaultDetectionCounter function	324

8.3.1.12	Event types class . . . . .	324
8.3.1.12.1	diag::DTCFormatType type . . . . .	325
8.3.1.12.2	diag::EventStatusBit type . . . . .	325
8.3.1.12.3	diag::EventStatusByte type . . . . .	326
8.3.1.12.4	diag::DebouncingState type . . . . .	326
8.3.1.13	MultipleEvent class . . . . .	326
8.3.1.13.1	diag::EventHandleType type . . . . .	327
8.3.1.13.2	diag::MultipleEvent::MultipleEvent type . . . . .	327
8.3.1.13.3	diag::MultipleEvent::Destructor type . . . . .	328
8.3.1.13.4	diag::MultipleEvent::GetEventStatus function . . . . .	328
8.3.1.13.5	diag::MultipleEvent::SetEventStatusChangedNotifier function . . . . .	328
8.3.1.13.6	diag::MultipleEvent::GetDTCNumber function . . . . .	329
8.3.1.13.7	diag::MultipleEvent::GetDebouncingStatus function . . . . .	330
8.3.1.13.8	diag::MultipleEvent::GetFaultDetectionCounter function . . . . .	330
8.3.1.14	DTCInformation class . . . . .	330
8.3.1.14.1	diag::ControlDtcStatusType type . . . . .	331
8.3.1.14.2	diag::UdsDtcStatusBitType type . . . . .	331
8.3.1.14.3	diag::DtcSuppressionType type . . . . .	332
8.3.1.14.4	diag::DTCInformation::UdsDtcStatusByteType type . . . . .	332
8.3.1.14.5	diag::DTCInformation::SnapshotDataIdenti- fierType type . . . . .	333
8.3.1.14.6	diag::DTCInformation::SnapshotDataRecord- Type type . . . . .	333
8.3.1.14.7	diag::DTCInformation::SnapshotRecordUp- datedType type . . . . .	334
8.3.1.14.8	diag::DTCInformation::DTCInformation function . . . . .	334
8.3.1.14.9	diag::DTCInformation::DTCInformation(DTCInformation&& move constructor . . . . .	334
8.3.1.14.10	diag::DTCInformation::DTCInformation(DTCInformation& copy constructor . . . . .	335
8.3.1.14.11	diag::DTCInformation::DTCInformation& oper- ator=(DTCInformation&&) move operator . . . . .	335
8.3.1.14.12	diag::DTCInformation::DTCInformation& oper- ator=(DTCInformation&) copy operator . . . . .	336
8.3.1.14.13	diag::DTCInformation::~~DTCInformation func- tion . . . . .	336
8.3.1.14.14	diag::DTCInformation::GetCurrentStatus func- tion . . . . .	336
8.3.1.14.15	diag::DTCInformation::GetDtcSuppression function . . . . .	337
8.3.1.14.16	diag::DTCInformation::SetDTCSta- tusChangedNotifier function . . . . .	337

8.3.1.14.17	diag::DTCInformation::SetDtcSuppression function . . . . .	338
8.3.1.14.18	diag::DTCInformation::SetSnap- shotRecordUpdatedNotifier function . . . . .	338
8.3.1.14.19	diag::DTCInformation::GetNumberOfStore- dEntries function . . . . .	339
8.3.1.14.20	diag::DTCInformation::SetNumberOfStore- dEntriesNotifier function . . . . .	339
8.3.1.14.21	diag::DTCInformation::Clear function . . . . .	340
8.3.1.14.22	diag::DTCInformation::GetControlDTCStatus function . . . . .	340
8.3.1.14.23	diag::DTCInformation::SetControlDtcStatus- Notifier function . . . . .	341
8.3.1.14.24	diag::DTCInformation::EnableControlDtc func- tion . . . . .	341
8.3.1.14.25	diag::DTCInformation::GetEventMemoryOver- flow function . . . . .	342
8.3.1.14.26	diag::DTCInformation::SetEventMemoryOver- flowNotifier function . . . . .	342
8.3.1.15	Conversation class . . . . .	342
8.3.1.15.1	diag::ActivityStatusType type . . . . .	343
8.3.1.15.2	diag::SessionControlType type . . . . .	343
8.3.1.15.3	diag::SessionControlType kDefaultSession . . . . .	344
8.3.1.15.4	diag::SessionControlType kExtendedDiagnos- ticSession . . . . .	344
8.3.1.15.5	diag::SessionControlType kProgrammingSes- sion . . . . .	344
8.3.1.15.6	diag::SessionControlType kSafetySystemDi- agnosticSession . . . . .	345
8.3.1.15.7	diag::SecurityLevelType type . . . . .	345
8.3.1.15.8	diag::SecurityLevelType kLocked . . . . .	345
8.3.1.15.9	diag::Conversation::ConversationIdentifierType type . . . . .	346
8.3.1.15.10	diag::Conversation::GetConversation function . . . . .	346
8.3.1.15.11	diag::Conversation::GetAllConversations func- tion . . . . .	346
8.3.1.15.12	diag::Conversation::GetCurrentActiveConver- sations function . . . . .	347
8.3.1.15.13	diag::Conversation::GetActivityStatus function . . . . .	347
8.3.1.15.14	diag::Conversation::SetActivityNotifier function . . . . .	348
8.3.1.15.15	diag::Conversation::GetConversationIdentifier function . . . . .	348
8.3.1.15.16	diag::Conversation::GetDiagnosticSession function . . . . .	349
8.3.1.15.17	diag::Conversation::SetDiagnosticSessionNo- tifier function . . . . .	349

8.3.1.15.18	diag::Conversation::GetDiagnosticSession-ShortName function . . . . .	349
8.3.1.15.19	diag::Conversation::GetDiagnosticSecurityLevel function . . . . .	350
8.3.1.15.20	diag::Conversation::SetSecurityLevelNotifier function . . . . .	350
8.3.1.15.21	diag::Conversation::GetDiagnosticSecurityLevelShortName function . . . . .	351
8.3.1.15.22	diag::Conversation::ResetToDefaultSession function . . . . .	351
8.3.1.16	Condition class . . . . .	351
8.3.1.16.1	diag::ConditionType type . . . . .	352
8.3.1.16.2	diag::Condition::Condition function . . . . .	352
8.3.1.16.3	diag::Condition::Condition(Condition&) move constructor . . . . .	353
8.3.1.16.4	diag::Condition::Condition(Condition&) copy constructor . . . . .	353
8.3.1.16.5	diag::Condition::Condition& operator=(Condition&) move operator . . . . .	353
8.3.1.16.6	diag::Condition::Condition& operator=(Condition&) copy operator . . . . .	354
8.3.1.16.7	diag::Condition::~~Condition function . . . . .	354
8.3.1.16.8	diag::Condition::GetCondition function . . . . .	354
8.3.1.16.9	diag::Condition::SetCondition function . . . . .	355
8.3.1.17	MultipleCondition class . . . . .	355
8.3.1.17.1	diag::ConditionHandleType type . . . . .	355
8.3.1.17.2	diag::MultipleCondition::MultipleCondition type . . . . .	356
8.3.1.17.3	diag::MultipleCondition::Destructor type . . . . .	356
8.3.1.17.4	diag::MultipleCondition::GetCondition function . . . . .	356
8.3.1.17.5	diag::MultipleCondition::SetCondition function . . . . .	357
8.3.1.18	OperationCycle class . . . . .	357
8.3.1.18.1	diag::OperationCycle::OperationCycle function . . . . .	358
8.3.1.18.2	diag::OperationCycle::OperationCycle(OperationCycle&) move constructor . . . . .	358
8.3.1.18.3	diag::OperationCycle::OperationCycle(OperationCycle&) copy constructor . . . . .	358
8.3.1.18.4	diag::OperationCycle::OperationCycle& operator=(OperationCycle&) move operator . . . . .	359
8.3.1.18.5	diag::OperationCycle::OperationCycle& operator=(OperationCycle&) copy operator . . . . .	359
8.3.1.18.6	diag::OperationCycle::~~OperationCycle function . . . . .	360
8.3.1.18.7	diag::OperationCycle::SetNotifier function . . . . .	360
8.3.1.18.8	diag::OperationCycle::RestartOperationCycle function . . . . .	360
8.3.1.19	Indicator class . . . . .	361
8.3.1.19.1	diag::IndicatorType type . . . . .	361

8.3.1.19.2	diag::Indicator::Indicator function . . . . .	362
8.3.1.19.3	diag::Indicator::Indicator(Indicator&&) move constructor . . . . .	362
8.3.1.19.4	diag::Indicator::Indicator(Indicator&) copy constructor . . . . .	363
8.3.1.19.5	diag::Indicator::Indicator& operator=(Indicator&&) move operator . . . . .	363
8.3.1.19.6	diag::Indicator::Indicator& operator=(Indicator&) copy operator . . . . .	363
8.3.1.19.7	diag::Indicator::~~Indicator function . . . . .	364
8.3.1.19.8	diag::Indicator::GetIndicator function . . . . .	364
8.3.1.19.9	diag::Indicator::SetNotifier function . . . . .	364
8.3.1.20	ServiceValidation class . . . . .	365
8.3.1.20.1	diag::ConfirmationStatusType . . . . .	365
8.3.1.20.2	diag::ServiceValidation::ServiceValidation function . . . . .	366
8.3.1.20.3	diag::ServiceValidation::ServiceValidation(ServiceValidation&&) move constructor . . . . .	366
8.3.1.20.4	diag::ServiceValidation::ServiceValidation(ServiceValidation&) copy constructor . . . . .	367
8.3.1.20.5	diag::ServiceValidation::ServiceValidation& operator=(ServiceValidation&&) move operator . . . . .	367
8.3.1.20.6	diag::ServiceValidation::ServiceValidation& operator=(ServiceValidation&) copy operator . . . . .	367
8.3.1.20.7	diag::ServiceValidation::~~ServiceValidation function . . . . .	368
8.3.1.20.8	diag::ServiceValidation::Validate function . . . . .	368
8.3.1.20.9	diag::ServiceValidation::Confirmation function . . . . .	371
8.3.1.20.10	diag::ServiceValidation::Offer function . . . . .	372
8.3.1.20.11	diag::ServiceValidation::StopOffer function . . . . .	372
8.3.1.21	SecurityAccess class . . . . .	372
8.3.1.21.1	diag::KeyCompareResultType type . . . . .	373
8.3.1.21.2	diag::SecurityAccess::SecurityAccess function . . . . .	373
8.3.1.21.3	diag::SecurityAccess::SecurityAccess(SecurityAccess&&) move constructor . . . . .	374
8.3.1.21.4	diag::SecurityAccess::SecurityAccess(SecurityAccess&) copy constructor . . . . .	374
8.3.1.21.5	diag::SecurityAccess::SecurityAccess& operator=(SecurityAccess&&) move operator . . . . .	374
8.3.1.21.6	diag::SecurityAccess::SecurityAccess& operator=(SecurityAccess&) copy operator . . . . .	375
8.3.1.21.7	diag::SecurityAccess::~~SecurityAccess function . . . . .	375
8.3.1.21.8	diag::SecurityAccess::GetSeed function . . . . .	375
8.3.1.21.9	diag::SecurityAccess::CompareKey function . . . . .	377
8.3.1.21.10	diag::SecurityAccess::Offer function . . . . .	378
8.3.1.21.11	diag::SecurityAccess::StopOffer function . . . . .	378



8.3.1.22	CommunicationControl class	379
8.3.1.22.1	diag::CommunicationControl::ComCtrlRequestParamsType type	379
8.3.1.22.2	diag::CommunicationControl::ComCtrlRequestParamsType-controlType type	379
8.3.1.22.3	diag::CommunicationControl::ComCtrlRequestParamsType-communicationType type	380
8.3.1.22.4	diag::CommunicationControl::ComCtrlRequestParamsType-nodeIdentificationNumber type	380
8.3.1.22.5	diag::CommunicationControl::CommunicationControl function	380
8.3.1.22.6	diag::CommunicationControl::CommunicationControl(CommunicationControl&&) move constructor	381
8.3.1.22.7	diag::CommunicationControl::CommunicationControl(CommunicationControl&) copy constructor	381
8.3.1.22.8	diag::CommunicationControl::CommunicationControl& operator=(CommunicationControl&&) move operator	382
8.3.1.22.9	diag::CommunicationControl::CommunicationControl& operator=(CommunicationControl&) copy operator	382
8.3.1.22.10	diag::CommunicationControl::~~CommunicationControl function	382
8.3.1.22.11	diag::CommunicationControl::CommCtrlRequest function	383
8.3.1.22.12	diag::CommunicationControl::Offer function	384
8.3.1.22.13	diag::CommunicationControl::StopOffer function	384
8.3.1.23	DownloadService class	384
8.3.1.23.1	diag::DownloadService::OperationOutput type	385
8.3.1.23.2	diag::DownloadService::DownloadService function	386
8.3.1.23.3	diag::DownloadService::DownloadService(DownloadService&&) move constructor	386
8.3.1.23.4	diag::DownloadService::DownloadService(DownloadService&) copy constructor	386
8.3.1.23.5	diag::DownloadService::DownloadService& operator=(DownloadService&&) move operator	387
8.3.1.23.6	diag::DownloadService::DownloadService& operator=(DownloadService&) copy operator	387
8.3.1.23.7	diag::DownloadService::~~DownloadService function	388
8.3.1.23.8	diag::DownloadService::RequestDownload function	388
8.3.1.23.9	diag::DownloadService::DownloadData function	389

8.3.1.23.10	diag::DownloadService::RequestDownloadExit function . . . . .	390
8.3.1.23.11	diag::DownloadService::Offer function . . . . .	391
8.3.1.23.12	diag::DownloadService::StopOffer function . . . . .	391
8.3.1.24	UploadService class . . . . .	392
8.3.1.24.1	diag::UploadService::OperationOutput type . . . . .	392
8.3.1.24.2	diag::UploadService::UploadService function . . . . .	393
8.3.1.24.3	diag::UploadService::UploadService(UploadService&& move constructor . . . . .	393
8.3.1.24.4	diag::UploadService::UploadService(UploadService& copy constructor . . . . .	393
8.3.1.24.5	diag::UploadService::UploadService& opera- tor=(UploadService&&) move operator . . . . .	394
8.3.1.24.6	diag::UploadService::UploadService& opera- tor=(UploadService&) copy operator . . . . .	394
8.3.1.24.7	diag::UploadService::~~UploadService function . . . . .	395
8.3.1.24.8	diag::UploadService::RequestUpload function . . . . .	395
8.3.1.24.9	diag::UploadService::UploadData function . . . . .	396
8.3.1.24.10	diag::UploadService::RequestUploadExit function . . . . .	397
8.3.1.24.11	diag::UploadService::Offer function . . . . .	398
8.3.1.24.12	diag::UploadService::StopOffer function . . . . .	399
8.3.1.24.13	diag::UploadService::OperationOutput re- sponseData variable . . . . .	399
8.3.1.25	EcuResetRequest class . . . . .	399
8.3.1.25.1	diag::EcuResetRequest::ResetRequestType type . . . . .	400
8.3.1.25.2	diag::EcuResetRequest::ResetRequestType kCustomReset . . . . .	400
8.3.1.25.3	diag::EcuResetRequest::ResetRequestType kHardReset . . . . .	400
8.3.1.25.4	diag::EcuResetRequest::ResetRequestType kKeyOffOnReset . . . . .	401
8.3.1.25.5	diag::EcuResetRequest::ResetRequestType kSoftReset . . . . .	401
8.3.1.25.6	diag::EcuResetRequest::StopOffer function . . . . .	401
8.3.1.25.7	diag::EcuResetRequest::Offer function . . . . .	402
8.3.1.25.8	diag::EcuResetRequest::RequestReset function . . . . .	402
8.3.1.25.9	diag::EcuResetRequest::ExecuteReset function . . . . .	405
8.3.1.25.10	diag::EcuResetRe- quest::EnableRapidShutdown function . . . . .	406
8.3.1.25.11	diag::EcuResetRequest::~~EcuResetRequest destructor . . . . .	409
8.3.1.25.12	diag::EcuResetRequest::EcuResetRequest constructor . . . . .	410

8.3.1.25.13	diag::EcuResetRequest::EcuResetRequest(EcuResetRequest&& move constructor	410
8.3.1.25.14	diag::EcuResetRequest::EcuResetRequest(EcuResetRequest&& copy constructor	410
8.3.1.25.15	diag::EcuResetRequest::EcuResetRequest& operator=(EcuResetRequest&&) move operator	411
8.3.1.25.16	diag::EcuResetRequest::EcuResetRequest& operator=(EcuResetRequest&) copy operator	411
8.3.1.26	Authentication class	412
8.3.1.26.1	diag::Authentication::Authentication constructor	412
8.3.1.26.2	diag::Authenticat- ion::Authentication(Authentication&&) move constructor	412
8.3.1.26.3	diag::Authenticat- ion::Authentication(Authentication&) copy constructor	413
8.3.1.26.4	diag::Authentication::Authentication& opera- tor=(Authentication&&) move operator	413
8.3.1.26.5	diag::Authentication::Authentication& opera- tor=(Authentication&) copy operator	414
8.3.1.26.6	diag::Authentication::~~Authentication destructor	414
8.3.1.26.7	diag::Authenticat- ion::VerifyCertificateUnidirectional function	414
8.3.1.26.8	diag::Authenticat- ion::VerifyCertificateBidirectional function	416
8.3.1.26.9	diag::Authentication::VerifyOwnership function	418
8.3.1.26.10	diag::Authentication::TransmitCertificate function	419
8.3.1.26.11	diag::Authentication::Offer function	420
8.3.1.26.12	diag::Authentication::StopOffer function	420
8.3.1.27	ClientAuthentication class	421
8.3.1.27.1	diag::ClientAuthentication::DiagnosticAuthState type	421
8.3.1.27.2	diag::ClientAuthentication::DiagnosticAuthRole type	421
8.3.1.27.3	diag::ClientAuthentication::~~ClientAuthentication destructor	422
8.3.1.27.4	diag::ClientAuthentication::ClientAuthentication copy constructor	422
8.3.1.27.5	diag::ClientAuthentication::ClientAuthentication move constructor	422
8.3.1.27.6	diag::ClientAuthentication::ClientAuthentication copy assignment operator	423
8.3.1.27.7	diag::ClientAuthentication::ClientAuthentication move assignment operator	423
8.3.1.27.8	diag::ClientAuthenticat- ion::OverrideDefaultRoles function	424

8.3.1.27.9	diag::ClientAuthentication::Authenticate function	424
8.3.1.27.10	diag::ClientAuthentication::GetState function	425
8.3.1.27.11	diag::ClientAuthentication::SetNotifier function	425
8.3.1.28	ClientAuthenticationHandle class	426
8.3.1.28.1	diag::ClientAuthenticationHandle::ClientAuthenticationHandle constructor	426
8.3.1.28.2	diag::ClientAuthenticationHandle::~~ClientAuthenticationHandle destructor	426
8.3.1.28.3	diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy constructor	427
8.3.1.28.4	diag::ClientAuthenticationHandle::ClientAuthenticationHandle move constructor	427
8.3.1.28.5	diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy assignment operator	428
8.3.1.28.6	diag::ClientAuthenticationHandle::ClientAuthenticationHandle move assignment operator	428
8.3.1.28.7	diag::ClientAuthenticationHandle::Append function	428
8.3.1.28.8	diag::ClientAuthenticationHandle::Set function	429
8.3.1.28.9	diag::ClientAuthenticationHandle::Revoke function	429
8.3.1.28.10	diag::ClientAuthenticationHandle::Refresh function	430
8.3.1.29	DiagnosticServiceDynamicAccessList class	430
8.3.1.29.1	diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList constructor	430
8.3.1.29.2	diag::DiagnosticServiceDynamicAccessList::~~DiagnosticServiceDynamicAccessList destructor	431
8.3.1.29.3	diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy constructor	431
8.3.1.29.4	diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move constructor	432
8.3.1.29.5	diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy assignment operator	432
8.3.1.29.6	diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move assignment operator	433
8.3.1.29.7	diag::DiagnosticServiceDynamicAccessList::Reserve function	433
8.3.1.29.8	diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function	434

8.3.1.29.9	diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function . . . . .	434
8.3.1.30	ExternalAuthentication class . . . . .	435
8.3.1.30.1	diag::ExternalAuthentication::Address type . . . . .	435
8.3.1.30.2	diag::ExternalAuthentication::ExternalAuthentication constructor . . . . .	435
8.3.1.30.3	diag::ExternalAuthentication::ExternalAuthentication copy constructor . . . . .	436
8.3.1.30.4	diag::ExternalAuthentication::ExternalAuthentication move constructor . . . . .	436
8.3.1.30.5	diag::ExternalAuthentication::ExternalAuthentication copy assignment operator . . . . .	437
8.3.1.30.6	diag::ExternalAuthentication::ExternalAuthentication move assignment operator . . . . .	437
8.3.1.30.7	diag::ExternalAuthentication::~~ExternalAuthentication destructor . . . . .	437
8.3.1.30.8	diag::ExternalAuthentication::Get function . . . . .	438
8.3.1.30.9	diag::ExternalAuthentication::Get function . . . . .	438
8.3.1.30.10	diag::ExternalAuthentication::GetAll function . . . . .	439
8.3.1.31	DynamicAccessListDiagServiceBuilder class . . . . .	439
8.3.1.31.1	ByteRange class . . . . .	439
8.3.1.31.1.1	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange constructor . . . . .	440
8.3.1.31.1.2	diag::DynamicAccessListDiagServiceBuilder::ByteRange::~~ByteRange destructor . . . . .	440
8.3.1.31.1.3	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy constructor . . . . .	441
8.3.1.31.1.4	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move constructor . . . . .	441
8.3.1.31.1.5	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy assignment operator . . . . .	442
8.3.1.31.1.6	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move assignment operator . . . . .	442
8.3.1.31.1.7	diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin function . . . . .	443
8.3.1.31.1.8	diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax function . . . . .	443

8.3.1.31.2	diag::DynamicAccessListDiagService-Builder::Byte type	443
8.3.1.31.3	diag::DynamicAccessListDiagService-Builder::ByteString type	444
8.3.1.31.4	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder constructor	444
8.3.1.31.5	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy constructor	445
8.3.1.31.6	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move constructor	445
8.3.1.31.7	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy assignment operator	445
8.3.1.31.8	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move assignment operator	446
8.3.1.31.9	diag::DynamicAccessListDiagServiceBuilder::~~DynamicAccessListDiagServiceBuilder destructor	446
8.3.1.31.10	diag::DynamicAccessListDiagService-Builder::Add function	447
8.3.1.31.11	diag::DynamicAccessListDiagService-Builder::Add function	447
8.3.1.31.12	diag::DynamicAccessListDiagService-Builder::Add function	448
8.3.1.31.13	diag::DynamicAccessListDiagService-Builder::Any function	448
8.3.1.31.14	diag::DynamicAccessListDiagService-Builder::EndsWith function	449
8.3.1.31.15	diag::DynamicAccessListDiagService-Builder::EndsWith function	449
8.3.1.31.16	diag::DynamicAccessListDiagService-Builder::Build function	450
8.3.1.32	FileTransferService class	450
8.3.1.32.1	diag::FileTransferService::FileSizes type	450
8.3.1.32.2	diag::FileTransferService::WriteFileMode enum	451
8.3.1.32.3	diag::FileTransferService::FileTransferService constructors	452
8.3.1.32.4	diag::FileTransferService::~~FileTransferService destructor	453
8.3.1.32.5	diag::FileTransferService::FileTransferService copy assignment operator	453

8.3.1.32.6	diag::FileTransferService::FileTransferService move assignment operator . . . . .	453
8.3.1.32.7	diag::FileTransferService::RequestReadFile function . . . . .	454
8.3.1.32.8	diag::FileTransferService::RequestReadDirec- tory function . . . . .	455
8.3.1.32.9	diag::FileTransferService::RequestWriteFile function . . . . .	455
8.3.1.32.10	diag::FileTransferService::RequestRe- sumeWriteFile function . . . . .	456
8.3.1.32.11	diag::FileTransferService::DeleteFile function .	457
8.3.1.32.12	diag::FileTransferService::Offer function . . . .	457
8.3.1.32.13	diag::FileTransferService::StopOffer function .	458
8.3.1.32.14	diag::DataTransferExitType enum . . . . .	458
8.3.1.33	DataTransferWriteHandler class . . . . .	459
8.3.1.33.1	diag::DataTransferWriteHandler::Data Trans- ferWriteHandler constructors . . . . .	459
8.3.1.33.2	diag::DataTransferWriteHandler::~~Data Trans- ferWriteHandler destructor . . . . .	460
8.3.1.33.3	diag::DataTransferWriteHandler::Data Trans- ferWriteHandler copy assignment operator . .	461
8.3.1.33.4	diag::DataTransferWriteHandler::Data Trans- ferWriteHandler move assignment operator . .	461
8.3.1.33.5	diag::DataTransferWriteHandler::Write function	461
8.3.1.33.6	diag::DataTransferWriteHandler::ExitWrite function . . . . .	462
8.3.1.34	DataTransferReadSharedDataHandler class . . . .	463
8.3.1.34.1	diag::DataTransferReadSharedDataHandler:: DataTransferReadSharedDataHandler con- structor . . . . .	463
8.3.1.34.2	diag::DataTransferReadSharedDataHandler:: DataTransferReadSharedDataHandler move constructor . . . . .	464
8.3.1.34.3	diag::DataTransferReadSharedDataHandler:: DataTransferReadSharedDataHandler copy constructor . . . . .	464
8.3.1.34.4	diag::DataTransferReadSharedDataHandler:: ~DataTransferReadSharedDataHandler de- structor . . . . .	464
8.3.1.34.5	diag::DataTransferReadSharedDataHandler:: DataTransferReadSharedDataHandler move assignment operator . . . . .	465
8.3.1.34.6	diag::DataTransferReadSharedDataHandler:: DataTransferReadSharedDataHandler copy assignment operator . . . . .	465

8.3.1.34.7	diag::DataTransferReadSharedDataHandler::Read function . . . . .	466
8.3.1.34.8	diag::DataTransferReadSharedDataHandler::ExitRead function . . . . .	466
8.3.1.35	DataTransferReadByPullHandler class . . . . .	467
8.3.1.35.1	diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler constructor . . . . .	468
8.3.1.35.2	diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler move constructor . . . . .	468
8.3.1.35.3	diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler copy constructor . . . . .	468
8.3.1.35.4	diag::DataTransferReadByPullHandler::~~DataTransferReadByPullHandler destructor . . . . .	469
8.3.1.35.5	diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler move assignment operator . . . . .	469
8.3.1.35.6	diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler copy assignment operator . . . . .	470
8.3.1.35.7	diag::DataTransferReadByPullHandler::Read function . . . . .	470
8.3.1.35.8	diag::DataTransferReadByPullHandler::ExitRead function . . . . .	471
8.3.1.36	DataTransferReadByPushHandler class . . . . .	472
8.3.1.36.1	diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler constructor . . . . .	472
8.3.1.36.2	diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler move constructor . . . . .	472
8.3.1.36.3	diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler copy constructor . . . . .	473
8.3.1.36.4	diag::DataTransferReadByPushHandler::~~DataTransferReadByPushHandler destructor . . . . .	473
8.3.1.36.5	diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler move assignment operator . . . . .	474
8.3.1.36.6	diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler copy assignment operator . . . . .	474
8.3.1.36.7	diag::DataTransferReadByPushHandler::Read function . . . . .	474
8.3.1.36.8	diag::DataTransferReadByPushHandler::ExitRead function . . . . .	475
8.3.1.37	DataTransferReadSession class . . . . .	476



8.3.1.37.1	diag::DataTransferReadSession::DataTransferReadSession constructors	476
8.3.1.37.2	diag::DataTransferReadSession::~~DataTransferReadSession destructor	477
8.3.1.37.3	diag::DataTransferReadSession::DataTransferReadSession copy assignment operator	478
8.3.1.37.4	diag::DataTransferReadSession::DataTransferReadSession move assignment operator	478
8.3.1.38	DataTransferWriteSession class	478
8.3.1.38.1	diag::DataTransferWriteSession::DataTransferWriteSession constructors	479
8.3.1.38.2	diag::DataTransferWriteSession::~~DataTransferWriteSession destructor	480
8.3.1.38.3	diag::DataTransferWriteSession::DataTransferWriteSession copy assignment operator	480
8.3.1.38.4	diag::DataTransferWriteSession::DataTransferWriteSession move assignment operator	480
8.3.1.39	ReleaseHandler class	481
8.3.1.39.1	diag::ReleaseHandler::ReleaseHandler constructors	481
8.3.1.39.2	diag::ReleaseHandler::ReleaseHandler copy assignment operator	482
8.3.1.39.3	diag::ReleaseHandler::ReleaseHandler move assignment operator	482
8.3.1.39.4	diag::ReleaseHandler::~~ReleaseHandler destructor	483
8.3.1.39.5	diag::ReleaseHandler::MayRelease function	483
8.3.1.39.6	diag::ReleaseHandler::SetNotifier function	484
8.3.2	C++ Diagnostic generated API Interfaces	484
8.3.2.1	Implementation Types header files	484
8.3.2.2	Typed Routine class	485
8.3.2.2.1	Routine::StartOutput type	486
8.3.2.2.2	Routine::StopOutput type	486
8.3.2.2.3	Routine::RequestResultsOutput type	486
8.3.2.2.4	Routine constructor	487
8.3.2.2.5	Routine move constructor	487
8.3.2.2.6	Routine copy constructor	488
8.3.2.2.7	Routine move operator	488
8.3.2.2.8	Routine copy operator	488
8.3.2.2.9	Routine destructor	489
8.3.2.2.10	Routine::Offer function	489
8.3.2.2.11	Routine::StopOffer function	489
8.3.2.2.12	Routine::Start function	490
8.3.2.2.13	Routine::Stop function	493
8.3.2.2.14	Routine::RequestResults function	497
8.3.2.3	Typed DataIdentifier class	500

	8.3.2.3.1	DataIdentifier::OperationOutput type . . . . .	501
	8.3.2.3.2	DataIdentifier constructor . . . . .	501
	8.3.2.3.3	DataIdentifier move constructor . . . . .	502
	8.3.2.3.4	DataIdentifier copy constructor . . . . .	502
	8.3.2.3.5	DataIdentifier move operator . . . . .	502
	8.3.2.3.6	DataIdentifier copy operator . . . . .	503
	8.3.2.3.7	DataIdentifier destructor . . . . .	503
	8.3.2.3.8	DataIdentifier::Offer function . . . . .	503
	8.3.2.3.9	DataIdentifier::StopOffer function . . . . .	504
	8.3.2.3.10	DataIdentifier::Read function . . . . .	504
	8.3.2.3.11	DataIdentifier::Write function . . . . .	507
8.3.2.4		Typed DataElement class . . . . .	511
	8.3.2.4.1	DataElement::OperationOutput type . . . . .	511
	8.3.2.4.2	DataElement constructor . . . . .	511
	8.3.2.4.3	DataElement move constructor . . . . .	512
	8.3.2.4.4	DataElement copy constructor . . . . .	512
	8.3.2.4.5	DataElement move operator . . . . .	513
	8.3.2.4.6	DataElement copy operator . . . . .	513
	8.3.2.4.7	DataElement destructor . . . . .	513
	8.3.2.4.8	DataElement::Offer function . . . . .	514
	8.3.2.4.9	DataElement::StopOffer function . . . . .	514
	8.3.2.4.10	DataElement::Read function . . . . .	514
8.3.3		C++ Diagnostic SOVD API Interfaces . . . . .	518
	8.3.3.1	SOVD Authorization . . . . .	518
	8.3.3.1.1	SovdAuthorization class . . . . .	518
	8.3.3.1.2	SovdAuthorization::SovdAuthorization constructor . . . . .	518
	8.3.3.1.3	SovdAuthorization::SovdAuthorization destructor . . . . .	519
	8.3.3.1.4	SovdAuthorization::GetTokenUrl function . . . . .	519
	8.3.3.1.5	SovdAuthorization::GetAuthorizationUrl function . . . . .	520
	8.3.3.1.6	SovdAuthorization::Offer function . . . . .	520
	8.3.3.1.7	SovdAuthorization::StopOffer function . . . . .	521
	8.3.3.1.8	SovdAuthorization::ValidateAuthorizationOutput type . . . . .	521
	8.3.3.1.9	SovdAuthorization::ValidateAuthorizationOutput identity . . . . .	521
	8.3.3.1.10	SovdAuthorization::ValidateAuthorizationOutput valid until . . . . .	522
	8.3.3.1.11	SovdAuthorization::ValidateAuthorization function . . . . .	522
8.3.3.2		SOVD Proximity Challenge . . . . .	523
	8.3.3.2.1	SovdProximityChallenge class . . . . .	523
	8.3.3.2.2	SovdProximityChallenge::SovdProximityChallengeType type . . . . .	523

8.3.3.2.3	SovdProximityChallenge::SovdProximityChallengeType valid until	524
8.3.3.2.4	SovdProximityChallenge::SovdProximityChallengeType challenge	524
8.3.3.2.5	SovdProximityChallenge::SovdProximityChallenge constructor	524
8.3.3.2.6	SovdProximityChallenge::SovdProximityChallenge destructor	525
8.3.3.2.7	SovdProximityChallenge::GetChallenge function	525
8.3.3.2.8	SovdProximityChallenge::Offer function	526
8.3.3.2.9	SovdProximityChallenge::StopOffer function	526
8.3.3.2.10	SovdProximityChallenge::ValidateResponse function	527
8.3.3.3	SOVD Service Validation	527
8.3.3.3.1	diag::SovdRequestMethod	527
8.3.3.3.2	SovdServiceValidation class	528
8.3.3.3.3	SovdServiceValidation::SovdServiceValidation constructor	528
8.3.3.3.4	SovdServiceValidation::SovdServiceValidation destructor	528
8.3.3.3.5	SovdServiceValidation::Offer function	529
8.3.3.3.6	SovdServiceValidation::StopOffer function	529
8.3.3.3.7	SovdServiceValidation::Validate function	530
8.3.3.4	Http Redirect type	530
8.3.3.4.1	HttpRedirect struct	530
8.3.3.4.2	HttpRedirect URL	531
8.3.3.5	SOVD Bulk Data	531
8.3.3.5.1	SovdBulkData	531
8.3.3.5.2	SovdBulkData::BulkDataMetaDataDescriptor type	532
8.3.3.5.3	SovdBulkData::DeleteByCategoryResult type	535
8.3.3.5.4	SovdBulkData::DeletionError type	536
8.3.3.5.5	SovdBulkData::SovdBulkData constructor	537
8.3.3.5.6	SovdBulkData::SovdBulkData destructor	538
8.3.3.5.7	SovdBulkData::copy assignment operator	538
8.3.3.5.8	SovdBulkData::move assignment operator	538
8.3.3.5.9	SovdBulkData::GetBulkDataMetaData function	539
8.3.3.5.10	SovdBulkData::RequestBulkDataDownload function	539
8.3.3.5.11	SovdBulkData::RequestBulkDataUpload function	540
8.3.3.5.12	SovdBulkData::DeleteAllBulkData function	541
8.3.3.5.13	SovdBulkData::DeleteSpecificBulkData function	541
8.3.3.5.14	SovdBulkData::Offer function	542
8.3.3.5.15	SovdBulkData::StopOffer function	542
8.3.3.6	SOVD Software Update	543

8.3.3.6.1	SovdSwUpdate	543
8.3.3.6.2	diag::SovdSwUpdate::SovdUpdatePhase enum	543
8.3.3.6.3	diag::SovdSwUpdate::SovdUpdateStatus enum	544
8.3.3.6.4	SovdSwUpdate::UpdatePackageDetails type	544
8.3.3.6.5	SovdSwUpdate::SubProgress type	549
8.3.3.6.6	SovdSwUpdate::UpdatePackageStatus type	550
8.3.3.6.7	SovdSwUpdate::SovdSwUpdate constructor	553
8.3.3.6.8	SovdSwUpdate::SovdSwUpdate destructor	554
8.3.3.6.9	SovdSwUpdate::copy assignment operator	554
8.3.3.6.10	SovdSwUpdate::move assignment operator	554
8.3.3.6.11	SovdSwUpdate::GetAllUpdates function	555
8.3.3.6.12	SovdSwUpdate::GetUpdatePackageDetails function	555
8.3.3.6.13	SovdSwUpdate::PutUpdatePackageAutomated function	556
8.3.3.6.14	SovdSwUpdate::PrepareUpdatePackage function	557
8.3.3.6.15	SovdSwUpdate::ExecuteUpdatePackage function	557
8.3.3.6.16	SovdSwUpdate::GetUpdatePackageStatus function	558
8.3.3.6.17	SovdSwUpdate::DeleteUpdatePackage function	558
8.3.3.6.18	SovdSwUpdate::RequestUpdatePackageRegistration function	559
8.3.3.6.19	SovdSwUpdate::Offer function	559
8.3.3.6.20	SovdSwUpdate::StopOffer function	560
8.3.3.7	SOVD Configuration	560
8.3.3.7.1	SovdConfiguration	560
8.3.3.7.2	SovdConfiguration::SovdConfigurationMetaInfo type	560
8.3.3.7.3	SovdConfiguration::SovdConfiguration constructor	561
8.3.3.7.4	SovdConfiguration::SovdConfiguration destructor	562
8.3.3.7.5	SovdConfiguration::copy assignment operator	563
8.3.3.7.6	SovdConfiguration::move assignment operator	563
8.3.3.7.7	SovdConfiguration::RequestGetConfiguration function	563
8.3.3.7.8	SovdConfiguration::RequestPutConfiguration function	564
8.3.3.7.9	SovdConfiguration::Offer function	565
8.3.3.7.10	SovdConfiguration::StopOffer function	565
A	Mentioned Manifest Elements	566
B	Platform Extension API (normative)	633
B.1	C++ UDS Transportlayer API Interfaces	633

B.1.1	UDS Transportlayer Types	633
B.1.1.1	uds_transport::ByteSpan	633
B.1.1.2	uds_transport::ChannelID	633
B.1.1.3	uds_transport::Priority	634
B.1.1.4	uds_transport::ProtocolKind	634
B.1.1.5	uds_transport::UdsMessageConstPtr	634
B.1.1.6	uds_transport::UdsMessagePtr	635
B.1.1.7	uds_transport::UdsTransportProtocolHandlerID	635
B.1.2	UdsMessage Class	635
B.1.2.1	UdsMessage Types	636
B.1.2.1.1	uds_transport::UdsMessage::Address	636
B.1.2.1.2	uds_transport::UdsMessage::MetaInfoMap	636
B.1.2.1.3	uds_transport::UdsMessage::TargetAddressType	637
B.1.2.2	UdsMessage Methods	637
B.1.2.2.1	uds_transport::UdsMessage::UdsMessage constructor	637
B.1.2.2.2	uds_transport::UdsMessage::~~UdsMessage	638
B.1.2.2.3	uds_transport::UdsMessage::AddMetaInfo	638
B.1.2.2.4	uds_transport::UdsMessage::GetPayload	638
B.1.2.2.5	uds_transport::UdsMessage::GetSa	639
B.1.2.2.6	uds_transport::UdsMessage::GetTa	640
B.1.2.2.7	uds_transport::UdsMessage::GetTaType	640
B.1.3	UdsTransportProtocolHandler Class	640
B.1.3.1	UdsTransportProtocolHandlerClass Types	641
B.1.3.1.1	uds_transport::UdsTransportProtocolHandler::InitializationResult	641
B.1.3.2	UdsTransportProtocolHandler Methods	641
B.1.3.2.1	uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler	641
B.1.3.2.2	uds_transport::UdsTransportProtocolHandler::~~UdsTransport	642
B.1.3.2.3	uds_transport::UdsTransportProtocolHandler::GetHandlerID	642
B.1.3.2.4	uds_transport::UdsTransportProtocolHandler::Initialize	643
B.1.3.2.5	uds_transport::UdsTransportProtocolHandler::NotifyReestablishment	643
B.1.3.2.6	uds_transport::UdsTransportProtocolHandler::Start	644
B.1.3.2.7	uds_transport::UdsTransportProtocolHandler::Stop	644
B.1.3.2.8	uds_transport::UdsTransportProtocolHandler::Transmit	645
B.1.3.2.9	uds_transport::UdsTransportProtocolHandler::GetPeriodicHandler	645

	B.1.3.2.10	uds_transport::UdsTransportProtocolHandler::transportProtocolM	variable	646
B.1.4		UdsTransportProtocolMgr Class		646
	B.1.4.1	UdsTransportProtocolMgrClass Types		647
		B.1.4.1.1	uds_transport::UdsTransportProtocolMgr::GlobalChannelIdentifier	647
		B.1.4.1.2	uds_transport::UdsTransportProtocolMgr::IndicationResult	647
		B.1.4.1.3	uds_transport::UdsTransportProtocolMgr::TransmissionResult	648
	B.1.4.2	UdsTransportProtocolMgr Methods		648
		B.1.4.2.1	uds_transport::UdsTransportProtocolMgr::~UdsTransportProtocolMgr	648
		B.1.4.2.2	uds_transport::UdsTransportProtocolMgr::ChannelReestablished	649
		B.1.4.2.3	uds_transport::UdsTransportProtocolMgr::HandleMessage	649
		B.1.4.2.4	uds_transport::UdsTransportProtocolMgr::HandlerStopped	650
		B.1.4.2.5	uds_transport::UdsTransportProtocolMgr::IndicateMessage	650
		B.1.4.2.6	uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure	651
		B.1.4.2.7	uds_transport::UdsTransportProtocolMgr::TransmitConfirmation	651
		B.1.4.2.8	uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation	652
B.1.5		UdsTransportProtocolPeriodicHandler Class		652
	B.1.5.1	UdsTransportProtocolPeriodicHandler Methods		653
		B.1.5.1.1	uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages	653
		B.1.5.1.2	uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength	653
		B.1.5.1.3	uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit	654
B.1.6		Sequence Diagrams of UDS Transport Layer Interaction		655
	B.1.6.1	Lifecycle		655
	B.1.6.2	UDS Request Processing		656
	B.1.6.3	UDS Response Transmission		657
	B.1.6.4	Channel Reestablishment		658
B.2		C++ DoIP API Interfaces		659
	B.2.1	DoIPGroupIdentification class		659
		B.2.1.1	diag::DoIPGroupIdentification::DoIPGroupIdentificationType type	659
		B.2.1.2	diag::DoIPGroupIdentification::DoIPGroupIdentification function	659

B.2.1.3	diag::DoIPGroupIdentification::DoIPGroupIdentification(DoIPGroupIdentification&&) move constructor . . . . .	660
B.2.1.4	diag::DoIPGroupIdentification::DoIPGroupIdentification(DoIPGroupIdentification&&) copy constructor . . . . .	660
B.2.1.5	diag::DoIPGroupIdentification::DoIPGroupIdentification& operator=(DoIPGroupIdentification&&) move operator	661
B.2.1.6	diag::DoIPGroupIdentification::DoIPGroupIdentification& operator=(DoIPGroupIdentification&) copy operator .	661
B.2.1.7	diag::DoIPGroupIdentification::~~DoIPGroupIdentification function . . . . .	661
B.2.1.8	diag::DoIPGroupIdentification::GetGidStatus function	662
B.2.1.9	diag::DoIPGroupIdentification::Offer function . . . . .	662
B.2.1.10	diag::DoIPGroupIdentification::StopOffer function . .	662
B.2.2	DoIPPowerMode class . . . . .	663
B.2.2.1	diag::PowerModeType type . . . . .	663
B.2.2.2	diag::DoIPPowerMode::DoIPPowerMode function . .	664
B.2.2.3	diag::DoIPPowerMode::DoIPPowerMode(DoIPPowerMode&&) move constructor . . . . .	664
B.2.2.4	diag::DoIPPowerMode::DoIPPowerMode(DoIPPowerMode&) copy constructor . . . . .	664
B.2.2.5	diag::DoIPPowerMode::DoIPPowerMode& operator=(DoIPPowerMode&&) move operator . . . . .	665
B.2.2.6	diag::DoIPPowerMode::DoIPPowerMode& operator=(DoIPPowerMode&) copy operator . . . . .	665
B.2.2.7	diag::DoIPPowerMode::~~DoIPPowerMode function .	666
B.2.2.8	diag::DoIPPowerMode::GetDoIPPowerMode function	666
B.2.2.9	diag::DoIPPowerMode::Offer function . . . . .	666
B.2.2.10	diag::DoIPPowerMode::StopOffer function . . . . .	667
B.2.3	DoIPActivationLine class . . . . .	667
B.2.3.1	diag::DoIPActivationLine::DoIPActivationLine function	668
B.2.3.2	diag::DoIPActivationLine::DoIPActivationLine(DoIPActivationLine&&) move constructor . . . . .	668
B.2.3.3	diag::DoIPActivationLine::DoIPActivationLine(DoIPActivationLine&) copy constructor . . . . .	668
B.2.3.4	diag::DoIPActivationLine::DoIPActivationLine& operator=(DoIPActivationLine&&) move operator . . . . .	669
B.2.3.5	diag::DoIPActivationLine::DoIPActivationLine& operator=(DoIPActivationLine&) copy operator . . . . .	669
B.2.3.6	diag::DoIPActivationLine::~~DoIPActivationLine function	670
B.2.3.7	diag::DoIPActivationLine::GetNetworkInterfaceId function . . . . .	670
B.2.3.8	diag::DoIPActivationLine::UpdateActivationLineState function . . . . .	670
B.2.3.9	diag::DoIPActivationLine::GetActivationLineState function . . . . .	671
B.2.3.10	diag::DoIPActivationLine::Offer function . . . . .	671

B.2.3.11	diag::DoIPActivationLine::StopOffer function . . . . .	672
B.2.4	DoIPTriggerVehicleAnnouncement class . . . . .	672
B.2.4.1	diag::DoIPTriggerVehicleAnnouncement::GetDoIP TriggerVehicleAnnouncement function . . . . .	672
B.2.4.2	diag::DoIPTriggerVehicleAnnouncement::TriggerVe- hicleAnnouncement function . . . . .	673
B.2.5	DoIPEntityIdentification class . . . . .	673
B.2.5.1	diag::DoIPEntityIdentification::DoIPEntityIdentifica- tionType type . . . . .	674
B.2.5.1.1	diag::DoIPEntityIdentification::EntityId entityI- dentification . . . . .	674
B.2.5.2	diag::DoIPEntityIdentification::DoIPEntityIdentifica- tion constructor . . . . .	675
B.2.5.3	diag::DoIPEntityIdentification::~~DoIPEntityIdentifica- tion destructor . . . . .	675
B.2.5.4	diag::DoIPEntityIdentification::GetEntityId function .	675
B.2.5.5	diag::DoIPEntityIdentification::Offer function . . . . .	676
B.2.5.6	diag::DoIPEntityIdentification::StopOffer function . .	676
C	Interfaces to other Functional Clusters (informative)	677
C.1	Overview . . . . .	677
C.2	Interface Tables . . . . .	677
D	History of Constraints and Specification Items	678
D.1	Constraint and Specification Item History of this document according to AUTOSAR Release 17-10 . . . . .	678
D.1.1	Added Specification Items in 17-10 . . . . .	678
D.1.2	Changed Specification Items in 17-10 . . . . .	680
D.1.3	Deleted Specification Items in 17-10 . . . . .	682
D.1.4	Added Constraints in 17-10 . . . . .	683
D.1.5	Changed Constraints in 17-10 . . . . .	683
D.1.6	Deleted Constraints in 17-10 . . . . .	683
D.2	Constraint and Specification Item History of this document according to AUTOSAR Release 18-03 . . . . .	684
D.2.1	Added Specification Items in 18-03 . . . . .	684
D.2.2	Changed Specification Items in 18-03 . . . . .	685
D.2.3	Deleted Specification Items in 18-03 . . . . .	691
D.2.4	Added Constraints in 18-03 . . . . .	691
D.2.5	Changed Constraints in 18-03 . . . . .	692
D.2.6	Deleted Constraints in 18-03 . . . . .	692
D.3	Constraint and Specification Item History of this document according to AUTOSAR Release 18-10 . . . . .	692
D.3.1	Added Specification Items in 18-10 . . . . .	692
D.3.2	Changed Specification Items in 18-10 . . . . .	694
D.3.3	Deleted Specification Items in 18-10 . . . . .	700
D.3.4	Added Constraints in 18-10 . . . . .	701
D.3.5	Changed Constraints in 18-10 . . . . .	701



D.3.6	Deleted Constraints in 18-10	702
D.4	Constraint and Specification Item History of this document according to AUTOSAR Release 19-03	702
D.4.1	Added Specification Items in 19-03	702
D.4.2	Changed Specification Items in 19-03	707
D.4.3	Deleted Specification Items in 19-03	708
D.4.4	Added Constraints in 19-03	708
D.4.5	Changed Constraints in 19-03	708
D.4.6	Deleted Constraints in 19-03	708
D.5	Constraint and Specification Item History of this document according to AUTOSAR Release R19-11	709
D.5.1	Added Specification Items in 19-11	709
D.5.2	Changed Specification Items in 19-11	712
D.5.3	Deleted Specification Items in 19-11	717
D.5.4	Added Constraints in 19-11	720
D.5.5	Changed Constraints in 19-11	720
D.5.6	Deleted Constraints in 19-11	720
D.6	Constraint and Specification Item History of this document according to AUTOSAR Release R20-11	720
D.6.1	Added Specification Items in R20-11	720
D.6.2	Changed Specification Items in R20-11	726
D.6.3	Deleted Specification Items in R20-11	729
D.6.4	Added Constraints in R20-11	730
D.6.5	Changed Constraints in R20-11	730
D.6.6	Deleted Constraints in R20-11	730
D.7	Constraint and Specification Item History of this document according to AUTOSAR Release R21-11	730
D.7.1	Added Specification Items in R21-11	730
D.7.2	Changed Specification Items in R21-11	735
D.7.3	Deleted Specification Items in R21-11	745
D.7.4	Added Constraints in R21-11	746
D.7.5	Changed Constraints in R21-11	746
D.7.6	Deleted Constraints in R21-11	746
D.8	Constraint and Specification Item History of this document according to AUTOSAR Release R22-11	746
D.8.1	Added Specification Items in R22-11	746
D.8.2	Changed Specification Items in R22-11	755
D.8.3	Deleted Specification Items in R22-11	768
D.8.4	Added Constraints in R22-11	769
D.8.5	Changed Constraints in R22-11	769
D.8.6	Deleted Constraints in R22-11	769
D.9	Constraint and Specification Item History of this document according to AUTOSAR Release R23-11	769
D.9.1	Added Specification Items in R23-11	769
D.9.2	Changed Specification Items in R23-11	780
D.9.3	Deleted Specification Items in R23-11	786

- D.9.4 Added Constraints in R23-11 . . . . . 786
- D.9.5 Changed Constraints in R23-11 . . . . . 787
- D.9.6 Deleted Constraints in R23-11 . . . . . 787

# 1 Introduction and functional overview

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

The [DM](#) is an [UDS](#) diagnostic implementation according to ISO 14229-1[1] and a [SOVD](#) implementation according [ASAMSOVD v 1.0.0 \[2\]](#) for the Autosar Adaptive Platform.

## 1.1 Diagnostic interface

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

## 1.2 AUTOSAR Diagnostic Extract Template (DEXT)

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

## 1.3 Software Cluster

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

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

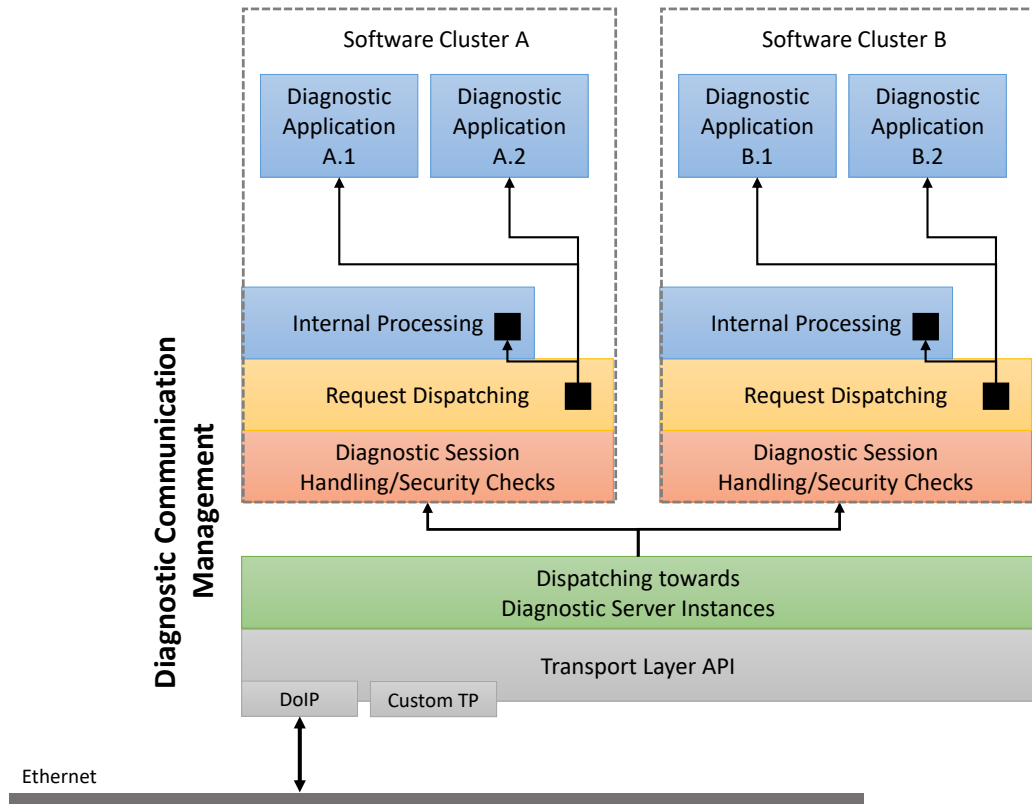
### 1.3.1 Diagnostic Server

The [Diagnostic Communication Management](#) response handling basically resembles the functionality of the [Dcm BSW](#) module of the AUTOSAR Classic platform. I.e. it is responsible for processing/dispatching of diagnostic services according to ISO 14229-1[1] and [SOVD](#) services as per Service-Oriented Vehicle Diagnostics standard Specification [2]. That means:

- Receiving diagnostic request messages from the network layer.
  - [UDS DoIP](#) or proprietary transport protocol requests
  - [SOVD HTTP/REST](#) requests

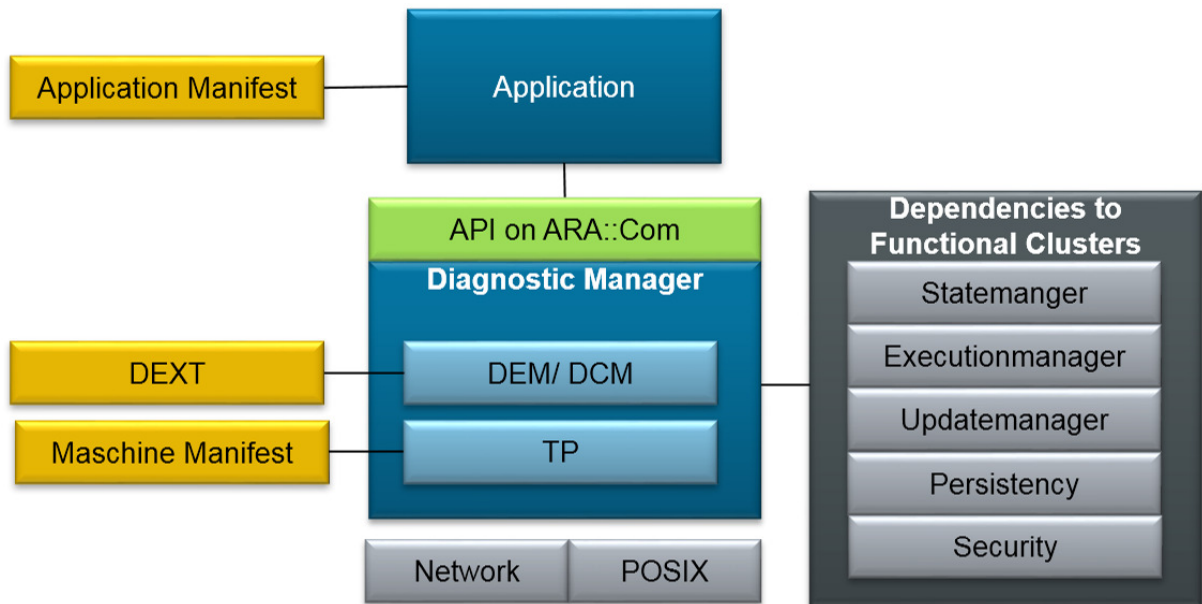
- Processing and extracting information from the request
  - transport layer independent UDS information
  - entity path and resource information for SOVD requests
- Dispatching the request towards the Diagnostic Server instances depending on
  - the target address and target address type (physical or 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 relevant to the [DM](#) module that are not included in the [\[4, AUTOSAR glossary\]](#).

Abbreviation / Acronym:	Description:
AA	AUTOSAR Adaptive Application
AP	AUTOSAR Adaptive Platform
API	Application Programming Interface
ASAM	Association for Standardization of Automation and Measuring Systems
BSW	Basic Software
Channel	An abstraction of a network specific communication channel. In CAN networks a Channel can be identified via CAN identifier. In Ethernet networks a Channel might be defined by the quadruple Src-IP, Src-Port, Target-IP, Target-Port.
CP	AUTOSAR Classic Platform
Dcm	Diagnostic Communication Manager (Module of the AUTOSAR Classic Platform)
DDDID	Dynamically Defined Data Identifier according to ISO 14229-1[1].
DEXT	AUTOSAR Diagnostic Extract[3], describing diagnostic configuration of an ECU
DID	Data Identifier according to ISO 14229-1[1]. This 16 bit value uniquely defines one or more data elements (parameters) that can be used in diagnostics to read, write or control data.
DM	AUTOSAR Adaptive Diagnostic Management
DNS	Domain Name System
DoIP	Diagnostics over Internet Protocol (Communication protocol of automotive electronics according to ISO-13400-2[5])
DTC	Diagnostic Trouble Code according to ISO 14229-1[1]
ECU	Electronic control unit
EDR	<a href="#">Extended Data Record</a>
EID	Entity Identification, as used in DoIP specification
Execution Management	Functional cluster Execution Management
FDC	Fault Detection Counter according to 14229-1[1]. Has always the value range from $-128 = FDC_{min} = \text{"FinallyHealed"}$ to $+127 = FDC_{max} = \text{"FinallyDefective"}$
GID	Group identifier as used in DoIP
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
MetaInfo	Meta-Information in the form of a key-value map, which is given from DM to external service processors.
NRC	Negative Response Code used by UDS in the diagnostic response to indicate the tester that a certain failure has occurred and the diagnostic request was not processed.
OBD	"On-Board Diagnostics"; Generally: A vehicles ability for self diagnosis and reporting to external test tools. Specifically here, the protocol is meant, as defined in SAE J1979, ISO 15031, ISO 27145 and others.
OEM	"Original Equipment Manufacturer", but in this document herein, it is used for "Vehicle Manufacturer".
PDID	Periodic Data Identifier according to ISO 14229-1[1].

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

Terms:	Description:
Aging	Unlearning/deleting of a no longer failed event/DTC after a defined number of operation cycles from event memory.
Diagnostic Client	A Diagnostic Client is a diagnostic service requester, i.e. sends a UDS request to the Diagnostic Server. Usually the Diagnostic Client is an external tester equipment but can also be another vehicle internal <a href="#">ECU</a> .
Diagnostic Communication Management	Diagnostic Communication Management is the part of the <a href="#">Diagnostic Management</a> which belongs to tester communication and the processing of UDS services.
Diagnostic Conversation	Diagnostic Conversation represents a conversation between Diagnostic Client (Tester) and Diagnostic Server.
Diagnostic Event Management	Diagnostic Event Management is the part of the <a href="#">Diagnostic Management</a> which belongs to processing and storing of diagnostic <a href="#">events</a> and associated data.
Diagnostic Management	Diagnostic Management is a placeholder for the complete functionality of diagnostic communication and event handling.
Diagnostic Server instance	Diagnostic Server ( <a href="#">DM</a> ) is intended to support an own Diagnostic Server instance per installed <i>SoftwareCluster</i> , see section 7.4 for a detailed description. Each of those Server instances has and manages its own resources and is responsible for dispatching and processing of diagnostic services.
Diagnostic Service instance	A diagnostic service instance implements a concrete use of a diagnostic service in a given context. It refers to a <a href="#">DiagnosticServiceClass</a> and the <a href="#">DiagnosticAccessPermission</a> , see 7.4.2.5.3 for a detailed description.
Displacement	In case of an <a href="#">Event memory overflow</a> : Replacing the most insignificant stored event memory entry by a reported event which needs to be stored and is more significant.
DTC group	Uniquely identifies a set of <a href="#">DTCs</a> . 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 <a href="#">UDS DTC status bits</a> are supported by the DM for masking purposes.



Terms:	Description:
Enable Conditions	The criteria / conditions under which the test results from the <a href="#">monitors</a> in the <a href="#">AA</a> 's are valid and shall be processed by <a href="#">DM</a> . Configuration is done per <a href="#">event</a> .
Extended Data Records	Contains statistical data for a DTC. Extended data records are assigned to DTCs and maintained and stored by the DM.
Event	An <i>event</i> (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.
Event status	Bit-packed status information based on <a href="#">Event</a> level. Contains the following bits:  Nr. Definition: <ul style="list-style-type: none"> <li>• 0 testFailed</li> <li>• 1 testFailedThisOperationCycle</li> <li>• 6 testNotCompletedThisOperationCycle</li> </ul> Compare <a href="#">UDS DTC status bit</a>
Fail-safe reaction	(Sometimes also called "limp home mode"): Reaction to avoid or minimize harm or damage in case of a failure.
GroupOfAllDTCs	Identifies a special DTC group that contains all DTCs. This DTC group is identified by the DTC value 0xFFFFFFFF in 14229-1[1] and contains by default all DTCs of a fault memory. It is present by default in the DM and requires no configuration.
Internal, External	Classifies if a <a href="#">DiagnosticDataElement</a> is either managed internally inside <a href="#">DM</a> or by an external adaptive applications, see <a href="#">7.4.6.1</a> 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 <a href="#">DiagnosticDataElement</a> which is provided by the DM itself. See also <a href="#">7.4.6.1</a> .
Monitor	A <i>monitor</i> (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 <a href="#">event</a> .
Operation cycle	A new operation cycle is the start of a new monitoring cycle. This is reflected in a reset of the testFailedThisOperationCycle and testNotCompletedThisOperationCycle bits in the DTC status and optional notification to the <a href="#">monitor</a> to restart the monitoring.
OpenAPI specification	Specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services.
Primary event memory	The primary event memory is used to store events and event related data. It is typically used by OEMs for after sales purposes, containing information to repair the vehicle.

Terms:	Description:
Snapshot Record	Snapshots (sometimes referred to as freeze frames ) are specific data records associated with a <a href="#">DTC</a> and stored in the fault memory at a certain point of time during fault detection. The <a href="#">DTC</a> specific data-parameters are intended to ease the fault isolation process.
SoftwareCluster	A SoftwareCluster groups all AUTOSAR artifacts which are relevant to deploy software on a machine. This includes the definition of applications, i.e. their executables, application manifests, communication and diagnostics. In the context of diagnostics a SoftwareCluster can be addressed individually by its own set of diagnostic addresses.
SourceAddress	A Source Address is used to encode client and server identifiers. In a UDS request the source address encodes the <a href="#">Diagnostic Client</a> whereas the source address in a UDS response encodes the Diagnostic Server.
SOVD entity	Entity in context of <a href="#">SOVD</a> [2]
SOVD lock	<a href="#">SOVD</a> locking mechanism as defined by [2]
TargetAddress	A Target Address is used to encode client and server identifiers. In a UDS request the target address encodes the Diagnostic Server whereas the target address in a UDS response encodes the <a href="#">Diagnostic Client</a> .
Transport Protocol Handler	A subcomponent of <a href="#">DM</a> implementing a particular Transport Protocol (either <a href="#">DoIP</a> or any other proprietary UDS Transport Layer).
Transport Protocol Manager	Link between UDS Transport Layer and Application Layer.
UDS service	A diagnostic service as defined in ISO 14229-1[1].
UDS DTC status bit	<p>UDS DTC status bit as defined in ISO 14229-1[1] Annex D.2; Each single bit position represents and documents a certain status information for the connected <a href="#">DTC</a>. The following eight bits are defined:</p> <p><b>Nr:</b> Definition:</p> <ul style="list-style-type: none"> <li><b>0</b> testFailed</li> <li><b>1</b> testFailedThisOperationCycle</li> <li><b>2</b> pendingDTC</li> <li><b>3</b> confirmedDTC</li> <li><b>4</b> testNotCompletedSinceLastClear</li> <li><b>5</b> testFailedSinceLastClear</li> <li><b>6</b> testNotCompletedThisOperationCycle</li> <li><b>7</b> warningIndicatorRequested</li> </ul> <p>All eight bits constitute the <a href="#">UDS DTC status byte</a>. Compare <a href="#">Event status</a></p>
UDS DTC status byte	Bit-packed DTC status information byte as defined in ISO 14229-1[1], based on DTC level. Contains the <a href="#">UDS DTC status bits</a> .
User-defined event memory	The user-defined event/fault memory is used by the UDS service 0x19 with subfunctions 0x17, 0x18 and 0x19. It behaves as the primary event memory but contains data independent from the primary fault memory. It is used to store information that are relevant for different purposes such as warranty or development.

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

## 3 Related documentation

### 3.1 Input documents & related standards and norms

- [1] Unified diagnostic services (UDS) – Part 1:Application layer (Release 2020-02)  
<https://www.iso.org>
- [2] ASAM SOVD Service-Oriented Vehicle Diagnostics - API Specification V1.0.0  
<http://www.asam.net>
- [3] Diagnostic Extract Template  
AUTOSAR\_CP\_TPS\_DiagnosticExtractTemplate
- [4] Glossary  
AUTOSAR\_FO\_TR\_Glossary
- [5] Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2:  
Network and transport layer requirements and services (Release 2019-12)  
<https://www.iso.org>
- [6] Specification of Adaptive Platform Core  
AUTOSAR\_AP\_SWS\_Core
- [7] Explanation of Adaptive Platform Software Architecture  
AUTOSAR\_AP\_EXP\_SWArchitecture
- [8] Requirements on Diagnostics  
AUTOSAR\_FO\_RS\_Diagnostics
- [9] Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2:Network  
layer services
- [10] Specification of Manifest  
AUTOSAR\_AP\_TPS\_ManifestSpecification
- [11] Specification of Intrusion Detection System Manager for Adaptive Platform  
AUTOSAR\_AP\_SWS\_IntrusionDetectionSystemManager
- [12] Unified diagnostic services (UDS) - Part 2:Session layer services (Release 2021-  
10)  
<https://www.iso.org>
- [13] Unified diagnostic services (UDS) – Part 1:Application layer (Release 2013-03)  
<https://www.iso.org>

### 3.2 Further applicable specification

AUTOSAR provides a core specification [6] which is also applicable for [Diagnostic Management](#). The chapter "General requirements for all FunctionalClusters" of this

specification shall be considered as an additional and required specification for implementation of [Diagnostic Management](#).

## 4 Constraints and assumptions

### 4.1 Known Limitations

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

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

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

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

For [SOVD\[2\]](#) 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.



## 5 Dependencies to other Functional Clusters

This chapter provides an overview of the dependencies to other Functional Clusters in the AUTOSAR Adaptive Platform. Section 5.1 “[Provided Interfaces](#)” lists the interfaces provided by Diagnostic Management to other Functional Clusters. Section 5.2 “[Required Interfaces](#)” lists the interfaces required by Diagnostic Management.

A detailed technical architecture documentation of the AUTOSAR Adaptive Platform is provided in [7].

### 5.1 Provided Interfaces

Interface	Functional Cluster	Purpose
No provided interfaces		

Table 5.1: Interfaces provided to other Functional Clusters

### 5.2 Required Interfaces

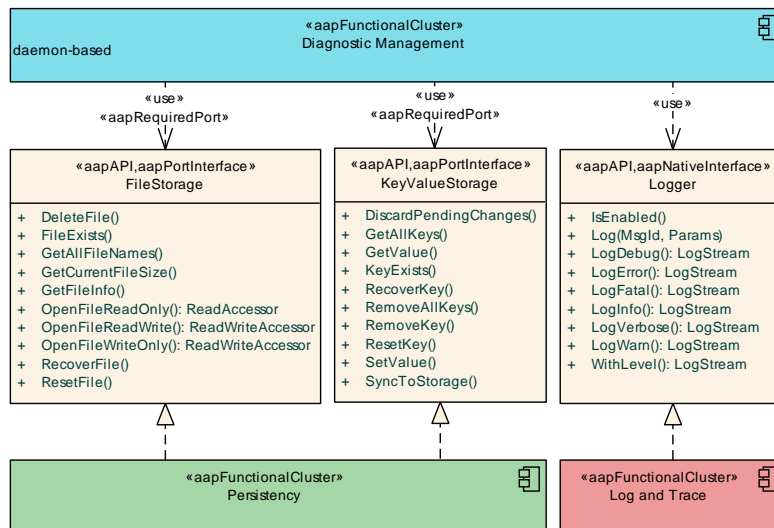
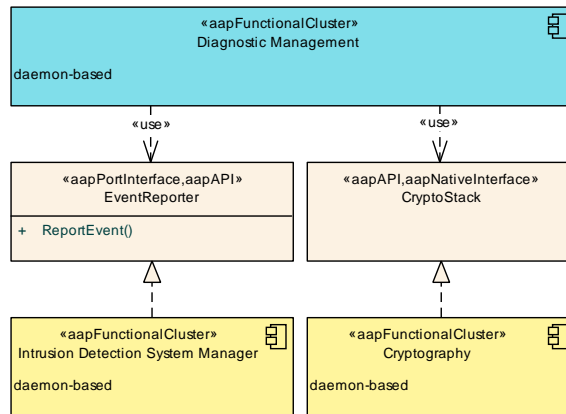
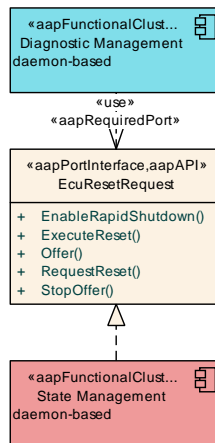


Figure 5.1: Interfaces required by Diagnostic Management



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



**Figure 5.3: Interfaces required by Diagnostic Management from State Management**

Figures 5.1, 5.2, and 5.3 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.
State Management	EcuResetRequest	This interface is used to handle reset requests.
Execution Management	ExecutionClient	This interface is used to report the status of the Diagnostic Management daemon process(es).
Intrusion Detection System Manager	EventReporter	Diagnostic Management uses this interface to report standardized security events.
Log and Trace	Logger	Diagnostic Management shall use this interface to log standardized messages.
Persistency	FileStorageOperations	This interface is used to read and write data to files.
Persistency	KeyValueStorageOperations	This interface should be used to persist key-value data.
Persistency	PersistencyHandlingOperations	This interface is used for general persistency handling.





<b>Functional Cluster</b>	<b>Interface</b>	<b>Purpose</b>
Persistency	FileStorage	This interface is used to read and write data to files.
Persistency	KeyValueStorage	This interface should be used to persist key-value data.
Persistency	ReadAccessor	This interface is used to read and write data to files.
Persistency	ReadWriteAccessor	This interface is used to read and write data to files.
Platform Health Management	SupervisedEntity	Diagnostic Management should use this interface to enable supervision of its daemon process(es) by Platform Health Management.

**Table 5.2: Interfaces required from other Functional Clusters**

## 6 Requirements Tracing

The following tables reference the requirements specified in [8] and links to the fulfilling requirements by this document. Please note that the column “Satisfied by” being empty for a specific requirement means that the requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_AP_00114]	C++ interface shall be compatible with C++14.	[SWS_DM_00510] [SWS_DM_00512] [SWS_DM_00513] [SWS_DM_00561]
[RS_AP_00116]	Header file name.	[SWS_DM_00511]
[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_00545] [SWS_DM_00546] [SWS_DM_00547] [SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00556] [SWS_DM_00557] [SWS_DM_00559] [SWS_DM_00560] [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_00618] [SWS_DM_00619] [SWS_DM_00636] [SWS_DM_00637] [SWS_DM_00638] [SWS_DM_00640] [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_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_01016] [SWS_DM_01088] [SWS_DM_01367] [SWS_DM_01698] [SWS_DM_01699] [SWS_DM_01709] [SWS_DM_01805] [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]





Requirement	Description	Satisfied by
[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]
[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_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_00128]	Error reporting.	[SWS_DM_00544] [SWS_DM_01126] [SWS_DM_01127] [SWS_DM_01128]
[RS_AP_00132]	noexcept behavior of API functions	[SWS_DM_00515] [SWS_DM_00516] [SWS_DM_00517] [SWS_DM_00518] [SWS_DM_00519] [SWS_DM_00520] [SWS_DM_00521] [SWS_DM_00522] [SWS_DM_00523] [SWS_DM_00524] [SWS_DM_00525] [SWS_DM_00527] [SWS_DM_00528] [SWS_DM_00529] [SWS_DM_00530] [SWS_DM_00531] [SWS_DM_00532] [SWS_DM_00533] [SWS_DM_00534] [SWS_DM_00535] [SWS_DM_00536] [SWS_DM_00537] [SWS_DM_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_01807] [SWS_DM_01808] [SWS_DM_01809] [SWS_DM_01810] [SWS_DM_01811] [SWS_DM_01812] [SWS_DM_01813] [SWS_DM_01814] [SWS_DM_01815] [SWS_DM_01816] [SWS_DM_01817]
[RS_AP_00133]	noexcept behavior of move and swap operations	[SWS_DM_00610] [SWS_DM_00612]





Requirement	Description	Satisfied by
[RS_AP_00134]	noexcept behavior of class destructors	[SWS_DM_00553] [SWS_DM_00584] [SWS_DM_00586] [SWS_DM_00588] [SWS_DM_00590] [SWS_DM_00635] [SWS_DM_00648] [SWS_DM_00665] [SWS_DM_00713] [SWS_DM_00723] [SWS_DM_00733] [SWS_DM_00743] [SWS_DM_00753] [SWS_DM_00763] [SWS_DM_00773] [SWS_DM_00788] [SWS_DM_00798] [SWS_DM_00807] [SWS_DM_00832] [SWS_DM_01365] [SWS_DM_01524] [SWS_DM_01706] [SWS_DM_01737]
[RS_AP_00137]	Connecting run-time interface with model.	[SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00552] [SWS_DM_00585] [SWS_DM_00587] [SWS_DM_00589] [SWS_DM_00616] [SWS_DM_00631] [SWS_DM_00633] [SWS_DM_00634] [SWS_DM_00647] [SWS_DM_00664] [SWS_DM_00712] [SWS_DM_00722] [SWS_DM_00732] [SWS_DM_00742] [SWS_DM_00752] [SWS_DM_00762] [SWS_DM_00772] [SWS_DM_00787] [SWS_DM_00797] [SWS_DM_00806] [SWS_DM_01364] [SWS_DM_01695] [SWS_DM_01696] [SWS_DM_01697] [SWS_DM_01702] [SWS_DM_01705] [SWS_DM_01727]
[RS_AP_00138]	Return type of asynchronous function calls.	[SWS_DM_00554] [SWS_DM_00555] [SWS_DM_00557] [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]
[RS_AP_00139]	Return type of synchronous function calls.	[SWS_DM_00543] [SWS_DM_00594] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00619] [SWS_DM_00638] [SWS_DM_00649] [SWS_DM_00650] [SWS_DM_00651] [SWS_DM_00652] [SWS_DM_00653] [SWS_DM_00654] [SWS_DM_00655] [SWS_DM_00656] [SWS_DM_00666] [SWS_DM_00667] [SWS_DM_00668] [SWS_DM_00669] [SWS_DM_00670] [SWS_DM_00671] [SWS_DM_00672] [SWS_DM_00673] [SWS_DM_00674] [SWS_DM_00692] [SWS_DM_00694] [SWS_DM_00695] [SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00700] [SWS_DM_00714] [SWS_DM_00715] [SWS_DM_00725] [SWS_DM_00735] [SWS_DM_00744] [SWS_DM_00745] [SWS_DM_00755] [SWS_DM_00766] [SWS_DM_00776] [SWS_DM_00792] [SWS_DM_00802]





Requirement	Description	Satisfied by
		<p style="text-align: center;">△</p> <p>[SWS_DM_00809] [SWS_DM_00918]            [SWS_DM_00919] [SWS_DM_01016]            [SWS_DM_01088] [SWS_DM_01102]            [SWS_DM_01292] [SWS_DM_01293]            [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]</p>
[RS_AP_00145]	Availability of special member functions.	<p>[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]</p>
[RS_AP_00146]	Classes whose construction requires interaction by the ARA framework.	<p>[SWS_DM_00609] [SWS_DM_00972]</p>
[RS_AP_00147]	Classes that are created with an InstanceSpecifier as an argument are not copyable, but at most movable.	<p>[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]</p>
[RS_AP_00149]	Guidance on error handling.	<p>[SWS_DM_01529]</p>





Requirement	Description	Satisfied by
[RS_Diag_04005]	Manage Security Access level handling	[SWS_DM_00047] [SWS_DM_00103] [SWS_DM_00236] [SWS_DM_00270] [SWS_DM_00271] [SWS_DM_00272] [SWS_DM_00421] [SWS_DM_00478] [SWS_DM_00479] [SWS_DM_00480] [SWS_DM_00760] [SWS_DM_00761] [SWS_DM_00762] [SWS_DM_00763] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00766] [SWS_DM_00767] [SWS_DM_01259] [SWS_DM_01260] [SWS_DM_01261] [SWS_DM_01758]
[RS_Diag_04006]	Manage session handling	[SWS_DM_00046] [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]
[RS_Diag_04016]	Support "Busy handling" by sending a negative response 0x78	[SWS_DM_00368] [SWS_DM_00369]
[RS_Diag_04019]	Provide confirmation after transmit diagnostic responses to the application	[SWS_DM_00268] [SWS_DM_00859]
[RS_Diag_04020]	Suppress responses to diagnostic tool requests	[SWS_DM_00433] [SWS_DM_01258]
[RS_Diag_04033]	Support the upload/download services for reading/writing data in an ECU in an extended and manufacturer specific diagnostic session	[SWS_DM_00128] [SWS_DM_00784] [SWS_DM_00785] [SWS_DM_00786] [SWS_DM_00787] [SWS_DM_00788] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00792] [SWS_DM_00793] [SWS_DM_00794] [SWS_DM_00795] [SWS_DM_00796] [SWS_DM_00797] [SWS_DM_00798] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00802] [SWS_DM_00803] [SWS_DM_00868] [SWS_DM_00869] [SWS_DM_00871] [SWS_DM_01096] [SWS_DM_01097]
[RS_Diag_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]







Requirement	Description	Satisfied by
[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_01100] [SWS_DM_01101] [SWS_DM_01268] [SWS_DM_01702] [SWS_DM_01710]
[RS_Diag_04071]	Process events according to their defined importance like priority and/or severity	[SWS_DM_00916] [SWS_DM_CONSTR_00961]
[RS_Diag_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]
[RS_Diag_04097]	Decentralized and modular diagnostic configuration in applications	[SWS_DM_00393] [SWS_DM_00401] [SWS_DM_00561] [SWS_DM_00570] [SWS_DM_00572] [SWS_DM_00848] [SWS_DM_00849] [SWS_DM_00850] [SWS_DM_00905] [SWS_DM_00906] [SWS_DM_00907] [SWS_DM_00908] [SWS_DM_01038] [SWS_DM_01039] [SWS_DM_01274] [SWS_DM_01275] [SWS_DM_CONSTR_00394] [SWS_DM_CONSTR_00395] [SWS_DM_CONSTR_00396]
[RS_Diag_04105]	Event memory management	[SWS_DM_00148] [SWS_DM_00150] [SWS_DM_00657] [SWS_DM_00664] [SWS_DM_00928] [SWS_DM_00929] [SWS_DM_00947] [SWS_DM_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]
[RS_Diag_04115]	The optional parameter DTCSetting ControlOptionRecord as part of UDS service ControlDTCSetting shall be limited to GroupOfDTC	[SWS_DM_00064] [SWS_DM_00231]
[RS_Diag_04117]	Configurable behavior for DTC deletion	[SWS_DM_00064] [SWS_DM_00065] [SWS_DM_00091] [SWS_DM_00092] [SWS_DM_00116] [SWS_DM_00117] [SWS_DM_00121] [SWS_DM_00122] [SWS_DM_00123] [SWS_DM_00124] [SWS_DM_00144] [SWS_DM_00145] [SWS_DM_00146] [SWS_DM_00147] [SWS_DM_00159] [SWS_DM_00160] [SWS_DM_00896] [SWS_DM_CONSTR_00082]
[RS_Diag_04118]	Optionally support event displacement	[SWS_DM_00916] [SWS_DM_00927] [SWS_DM_00928] [SWS_DM_00929] [SWS_DM_00930] [SWS_DM_00932] [SWS_DM_00933] [SWS_DM_00934] [SWS_DM_01569] [SWS_DM_CONSTR_00961]





Requirement	Description	Satisfied by
[RS_Diag_04119]	Handle the execution of diagnostic services according to the assigned diagnostic session	[SWS_DM_00046]
[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_00088] [SWS_DM_00378] [SWS_DM_00875] [SWS_DM_00876] [SWS_DM_00881] [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_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_CONSTR_00960]
[RS_Diag_04135]	Support UDS service \$38 (Request FileTransfer)	[SWS_DM_01310] [SWS_DM_01311] [SWS_DM_01312] [SWS_DM_01313] [SWS_DM_01314] [SWS_DM_01315] [SWS_DM_01316] [SWS_DM_01317] [SWS_DM_01318] [SWS_DM_01319] [SWS_DM_01320] [SWS_DM_01321] [SWS_DM_01322] [SWS_DM_01323] [SWS_DM_01324] [SWS_DM_01325] [SWS_DM_01326] [SWS_DM_01327] [SWS_DM_01328] [SWS_DM_01329] [SWS_DM_01330] [SWS_DM_01331] [SWS_DM_01332] [SWS_DM_01333] [SWS_DM_01334] [SWS_DM_01335] [SWS_DM_01336] [SWS_DM_01337] [SWS_DM_01339] [SWS_DM_01340] [SWS_DM_01497] [SWS_DM_01498] [SWS_DM_01499] [SWS_DM_01500] [SWS_DM_01501] [SWS_DM_01502] [SWS_DM_01503] [SWS_DM_01504] [SWS_DM_01505] [SWS_DM_01506] [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]





Requirement	Description	Satisfied by
		<p style="text-align: center;">△</p> <p>[SWS_DM_01553] [SWS_DM_01554]            [SWS_DM_01555] [SWS_DM_01556]            [SWS_DM_01557] [SWS_DM_01558]            [SWS_DM_01559] [SWS_DM_01560]</p>
[RS_Diag_04136]	Configurable "confirmed" threshold	[SWS_DM_00218]
[RS_Diag_04140]	Aging for UDS status bits "confirmed DTC" and "testFailedSinceLastClear"	[SWS_DM_00243]
[RS_Diag_04147]	Communication with the transport layers to receive and send diagnostic data	<p>[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_00451]            [SWS_DM_00452] [SWS_DM_09010]            [SWS_DM_09012] [SWS_DM_09015]            [SWS_DM_09016] [SWS_DM_09017]            [SWS_DM_09021]</p>
[RS_Diag_04148]	Provide capabilities to inform applications about diagnostic data changes	[SWS_DM_00667] [SWS_DM_00894]
[RS_Diag_04150]	Support the primary fault memory defined by ISO 14229-1	<p>[SWS_DM_00055] [SWS_DM_00056]            [SWS_DM_00083] [SWS_DM_00657]            [SWS_DM_00664] [SWS_DM_00911]            [SWS_DM_CONSTR_00084]</p>
[RS_Diag_04151]	Event status handling	<p>[SWS_DM_00218] [SWS_DM_00643]            [SWS_DM_00644] [SWS_DM_00646]            [SWS_DM_00647] [SWS_DM_00648]            [SWS_DM_00649] [SWS_DM_00652]            [SWS_DM_00655] [SWS_DM_00658]            [SWS_DM_00659] [SWS_DM_01024]            [SWS_DM_01025] [SWS_DM_01026]            [SWS_DM_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]</p>
[RS_Diag_04157]	Reporting of DTCs and related data	<p>[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]</p>





Requirement	Description	Satisfied by
[RS_Diag_04159]	Control of DTC storage	[SWS_DM_00088] [SWS_DM_00229] [SWS_DM_00378] [SWS_DM_00663] [SWS_DM_00672] [SWS_DM_00673] [SWS_DM_00674] [SWS_DM_00811] [SWS_DM_00909] [SWS_DM_00910] [SWS_DM_01094] [SWS_DM_01095] [SWS_DM_01353]
[RS_Diag_04160]	ResponseOnEvent according to ISO 14229-1	[SWS_DM_00491] [SWS_DM_00493] [SWS_DM_00494] [SWS_DM_01098] [SWS_DM_01117] [SWS_DM_01118] [SWS_DM_01119] [SWS_DM_01120] [SWS_DM_01121] [SWS_DM_01122] [SWS_DM_01262] [SWS_DM_01263]
[RS_Diag_04165]	Triggering of multiple events upon a master event is reported	[SWS_DM_01733]
[RS_Diag_04166]	Several tester conversations in parallel with assigned priorities	[SWS_DM_00426] [SWS_DM_00427] [SWS_DM_00428] [SWS_DM_00429] [SWS_DM_00430] [SWS_DM_00690] [SWS_DM_00691] [SWS_DM_00693] [SWS_DM_00701] [SWS_DM_00782] [SWS_DM_00783] [SWS_DM_00840] [SWS_DM_00841] [SWS_DM_00843] [SWS_DM_00844] [SWS_DM_00856] [SWS_DM_00935] [SWS_DM_00936] [SWS_DM_00937] [SWS_DM_00938] [SWS_DM_00939] [SWS_DM_00940] [SWS_DM_00941] [SWS_DM_00942] [SWS_DM_00943] [SWS_DM_00944] [SWS_DM_01581]
[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_00452] [SWS_DM_00487] [SWS_DM_01741] [SWS_DM_01742] [SWS_DM_01743] [SWS_DM_01744] [SWS_DM_01745] [SWS_DM_01746] [SWS_DM_09015] [SWS_DM_09016] [SWS_DM_09017] [SWS_DM_09021]





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_00579] [SWS_DM_00580] [SWS_DM_00581] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00584] [SWS_DM_00585] [SWS_DM_00586] [SWS_DM_00587] [SWS_DM_00588] [SWS_DM_00589] [SWS_DM_00590] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00594] [SWS_DM_00595] [SWS_DM_00596] [SWS_DM_00597] [SWS_DM_00598] [SWS_DM_00599] [SWS_DM_00600] [SWS_DM_00601] [SWS_DM_00602] [SWS_DM_00603] [SWS_DM_00604] [SWS_DM_00605] [SWS_DM_00607] [SWS_DM_00608] [SWS_DM_00609] [SWS_DM_00610] [SWS_DM_00611] [SWS_DM_00612] [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_00640] [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]





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_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_00692] [SWS_DM_00764] [SWS_DM_00765] [SWS_DM_00774] [SWS_DM_00775] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00808] [SWS_DM_00971] [SWS_DM_00972] [SWS_DM_00973] [SWS_DM_00974] [SWS_DM_00975] [SWS_DM_00976] [SWS_DM_00977] [SWS_DM_00978] [SWS_DM_00979] [SWS_DM_00980] [SWS_DM_01342] [SWS_DM_01343] [SWS_DM_01344] [SWS_DM_01345] [SWS_DM_01818]
[RS_Diag_04172]	Inform external service processors about outcome of the final response	[SWS_DM_00307] [SWS_DM_00312] [SWS_DM_00578] [SWS_DM_00618] [SWS_DM_00632] [SWS_DM_00636] [SWS_DM_00641] [SWS_DM_00859]
[RS_Diag_04173]	Different signature types, when delegating processing of UDS service to the application	[SWS_DM_00618]
[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_CONSTR_00168]
[RS_Diag_04179]	Provide interfaces for monitoring application.	[SWS_DM_00540] [SWS_DM_00541] [SWS_DM_00542] [SWS_DM_00543] [SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00965] [SWS_DM_01694] [SWS_DM_01695] [SWS_DM_01696] [SWS_DM_01697] [SWS_DM_01698] [SWS_DM_01702] [SWS_DM_01732]
[RS_Diag_04180]	Process all UDS Services related to diagnostic fault memory of ISO 14229-1 internally	[SWS_DM_00062] [SWS_DM_00090] [SWS_DM_00091] [SWS_DM_00092] [SWS_DM_00115] [SWS_DM_00162] [SWS_DM_00163] [SWS_DM_00164] [SWS_DM_00217] [SWS_DM_00229] [SWS_DM_00244] [SWS_DM_00245] [SWS_DM_00246] [SWS_DM_00247] [SWS_DM_00370] [SWS_DM_00371] [SWS_DM_00372] [SWS_DM_00373] [SWS_DM_00374] [SWS_DM_00909] [SWS_DM_00910] [SWS_DM_00966] [SWS_DM_00967] [SWS_DM_00968] [SWS_DM_01028] [SWS_DM_01256]





Requirement	Description	Satisfied by
[RS_Diag_04182]	Provide an application interface to change operation cycles states	[SWS_DM_01027] [SWS_DM_01102] [SWS_DM_01103]
[RS_Diag_04183]	Notify interested parties about event status changes	[SWS_DM_00650] [SWS_DM_00886] [SWS_DM_01029] [SWS_DM_01030] [SWS_DM_01031] [SWS_DM_01350] [SWS_DM_01351] [SWS_DM_01708]
[RS_Diag_04185]	Notify applications about the clearing of an event	[SWS_DM_00562]
[RS_Diag_04186]	Notify applications about the start or restart of an operation cycle	[SWS_DM_00563] [SWS_DM_00755] [SWS_DM_01358]
[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]
[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_00881] [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_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]
[RS_Diag_04196]	UDS Service handling for all diagnostic services defined in ISO 14229-2	[SWS_DM_00090] [SWS_DM_00096] [SWS_DM_00097] [SWS_DM_00113] [SWS_DM_00126] [SWS_DM_00127] [SWS_DM_00128] [SWS_DM_00134] [SWS_DM_00137] [SWS_DM_00140] [SWS_DM_00141] [SWS_DM_00162] [SWS_DM_00170] [SWS_DM_00186] [SWS_DM_00199] [SWS_DM_00201] [SWS_DM_00227] [SWS_DM_00234] [SWS_DM_00235] [SWS_DM_00236] [SWS_DM_00269] [SWS_DM_00360] [SWS_DM_00363] [SWS_DM_00571] [SWS_DM_00573] [SWS_DM_00574] [SWS_DM_00575] [SWS_DM_00576] [SWS_DM_00784] [SWS_DM_00785] [SWS_DM_00786] [SWS_DM_00787] [SWS_DM_00788] [SWS_DM_00789] [SWS_DM_00790] [SWS_DM_00791] [SWS_DM_00792] [SWS_DM_00793] [SWS_DM_00794] [SWS_DM_00795] [SWS_DM_00796] [SWS_DM_00797]





Requirement	Description	Satisfied by
		<p>△</p> <p>[SWS_DM_00798] [SWS_DM_00799]            [SWS_DM_00800] [SWS_DM_00801]            [SWS_DM_00802] [SWS_DM_00803]            [SWS_DM_00804] [SWS_DM_00805]            [SWS_DM_00806] [SWS_DM_00807]            [SWS_DM_00808] [SWS_DM_00809]            [SWS_DM_00810] [SWS_DM_00860]            [SWS_DM_00866] [SWS_DM_00867]            [SWS_DM_01007] [SWS_DM_01009]            [SWS_DM_01010] [SWS_DM_01011]            [SWS_DM_01012] [SWS_DM_01013]            [SWS_DM_01014] [SWS_DM_01017]            [SWS_DM_01018] [SWS_DM_01019]            [SWS_DM_01020] [SWS_DM_01021]            [SWS_DM_01022] [SWS_DM_01023]            [SWS_DM_01090] [SWS_DM_01092]            [SWS_DM_01258] [SWS_DM_01270]            [SWS_DM_01271] [SWS_DM_01272]            [SWS_DM_01273] [SWS_DM_01309]            [SWS_DM_01346] [SWS_DM_01347]            [SWS_DM_01562] [SWS_DM_01563]            [SWS_DM_01564]</p>
[RS_Diag_04197]	Clearing the user defined fault memory	[SWS_DM_00193] [SWS_DM_00194] [SWS_DM_00195] [SWS_DM_00208]
[RS_Diag_04198]	Process all UDS Services related to session and security management of ISO 14229 internally	[SWS_DM_00226] [SWS_DM_00228]
[RS_Diag_04199]	Provide a configurable UDS service execution mechanism at runtime to decide if a UDS request shall be processed or not	[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]
[RS_Diag_04201]	Support a configuration to assign specific events to a customer specific DTC	[SWS_DM_00060] [SWS_DM_00642] [SWS_DM_00653] [SWS_DM_01709] [SWS_DM_01733] [SWS_DM_CONSTR_00059]







Requirement	Description	Satisfied by
[RS_Diag_04203]	Common checks on all supported UDS Services Requests	[SWS_DM_00096] [SWS_DM_00098] [SWS_DM_00099] [SWS_DM_00100] [SWS_DM_00101] [SWS_DM_00102] [SWS_DM_00103] [SWS_DM_00202] [SWS_DM_00203] [SWS_DM_00230] [SWS_DM_00231] [SWS_DM_00252] [SWS_DM_00409] [SWS_DM_00412] [SWS_DM_00413] [SWS_DM_00414] [SWS_DM_00415] [SWS_DM_00416] [SWS_DM_00417] [SWS_DM_00437] [SWS_DM_00444] [SWS_DM_00445] [SWS_DM_00448] [SWS_DM_00450] [SWS_DM_00507] [SWS_DM_00863] [SWS_DM_00864]
[RS_Diag_04204]	Provide the current status of each warning indicator.	[SWS_DM_00223] [SWS_DM_00224] [SWS_DM_00651] [SWS_DM_00740] [SWS_DM_00741] [SWS_DM_00742] [SWS_DM_00743] [SWS_DM_00744] [SWS_DM_00745] [SWS_DM_00888] [SWS_DM_01032] [SWS_DM_01033] [SWS_DM_01034] [SWS_DM_01035] [SWS_DM_01266] [SWS_DM_01359] [SWS_DM_01582]
[RS_Diag_04205]	Support of SnapshotRecords	[SWS_DM_00151] [SWS_DM_00152] [SWS_DM_00660] [SWS_DM_00661] [SWS_DM_00662] [SWS_DM_00668] [SWS_DM_00969] [SWS_DM_01085] [SWS_DM_01086] [SWS_DM_01087] [SWS_DM_01276] [SWS_DM_01277]
[RS_Diag_04206]	Support of ExtendedDataRecords	[SWS_DM_00154] [SWS_DM_00155] [SWS_DM_00895]
[RS_Diag_04208]	Inform the application about diagnostic session and diagnostic security level changes on each tester connection.	[SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00707] [SWS_DM_00708] [SWS_DM_00845] [SWS_DM_00846] [SWS_DM_01356] [SWS_DM_01357]
[RS_Diag_04209]	Pseudo parallel client interaction according to ISO	[SWS_DM_00690] [SWS_DM_00691] [SWS_DM_00701] [SWS_DM_00782] [SWS_DM_00783]
[RS_Diag_04211]	Persistent storage of DTC status and environmental data	[SWS_DM_00148] [SWS_DM_00150] [SWS_DM_00811]
[RS_Diag_04214]	Support the user defined fault memories defined by ISO 14229-1	[SWS_DM_00055] [SWS_DM_00057] [SWS_DM_00083] [SWS_DM_00911] [SWS_DM_CONSTR_00084]
[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]





Requirement	Description	Satisfied by
[RS_Diag_04216]	Support for multiple Diagnostic Server Instances	[SWS_DM_00390] [SWS_DM_00391] [SWS_DM_00420]
[RS_Diag_04220]	Support DTCs suppression	[SWS_DM_01292] [SWS_DM_01293] [SWS_DM_01294] [SWS_DM_01296] [SWS_DM_01297] [SWS_DM_01298] [SWS_DM_01299] [SWS_DM_01300] [SWS_DM_01301] [SWS_DM_01302] [SWS_DM_01303] [SWS_DM_01304] [SWS_DM_01305] [SWS_DM_01306] [SWS_DM_01307] [SWS_DM_01308]
[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_00581] [SWS_DM_00582] [SWS_DM_00583] [SWS_DM_00591] [SWS_DM_00592] [SWS_DM_00593] [SWS_DM_00594] [SWS_DM_00605] [SWS_DM_00633] [SWS_DM_01270] [SWS_DM_01271] [SWS_DM_01272] [SWS_DM_01273]
[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_01697] [SWS_DM_01710]
[RS_Diag_04233]	Access granularity of diagnostic services	[SWS_DM_01223] [SWS_DM_01224] [SWS_DM_01225]
[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]
[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]





Requirement	Description	Satisfied by
[RS_Diag_04246]	Support of UDS service Dynamically DefineDataIdentifier (0x2C) with subfunction 0x01 (defineByIdentifier)	[SWS_DM_01070] [SWS_DM_01071] [SWS_DM_01072] [SWS_DM_01073] [SWS_DM_01074] [SWS_DM_01075] [SWS_DM_01076] [SWS_DM_01077] [SWS_DM_01078] [SWS_DM_01079] [SWS_DM_01080] [SWS_DM_01081] [SWS_DM_01082] [SWS_DM_01083]
[RS_Diag_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_01129] [SWS_DM_01130] [SWS_DM_01131] [SWS_DM_01132] [SWS_DM_01133] [SWS_DM_01134] [SWS_DM_01136] [SWS_DM_01137] [SWS_DM_01138] [SWS_DM_01139] [SWS_DM_01140] [SWS_DM_01141] [SWS_DM_01142] [SWS_DM_01143] [SWS_DM_01144] [SWS_DM_01145] [SWS_DM_01146] [SWS_DM_01147] [SWS_DM_01148] [SWS_DM_01149] [SWS_DM_01150] [SWS_DM_01151] [SWS_DM_01152] [SWS_DM_01153] [SWS_DM_01154] [SWS_DM_01155] [SWS_DM_01156] [SWS_DM_01157] [SWS_DM_01158] [SWS_DM_01159] [SWS_DM_01160] [SWS_DM_01161] [SWS_DM_01162] [SWS_DM_01163] [SWS_DM_01164] [SWS_DM_01165] [SWS_DM_01166] [SWS_DM_01167] [SWS_DM_01168] [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_01234] [SWS_DM_01235] [SWS_DM_01236] [SWS_DM_01238] [SWS_DM_01239] [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]





Requirement	Description	Satisfied by
[RS_Diag_04256]	Support of SOVD	[SWS_DM_01369] [SWS_DM_01370] [SWS_DM_01371] [SWS_DM_01372] [SWS_DM_01373] [SWS_DM_01374] [SWS_DM_01450] [SWS_DM_01451] [SWS_DM_01452] [SWS_DM_01453] [SWS_DM_01454] [SWS_DM_01455] [SWS_DM_01457] [SWS_DM_01458] [SWS_DM_01459] [SWS_DM_01460] [SWS_DM_01461] [SWS_DM_01462] [SWS_DM_01463] [SWS_DM_01464] [SWS_DM_01465] [SWS_DM_01466] [SWS_DM_01467] [SWS_DM_01468] [SWS_DM_01821] [SWS_DM_01822] [SWS_DM_01823] [SWS_DM_01824] [SWS_DM_01825] [SWS_DM_01826] [SWS_DM_01827] [SWS_DM_01828] [SWS_DM_01829]
[RS_Diag_04257]	Capability Description	[SWS_DM_01456] [SWS_DM_01786]
[RS_Diag_04258]	Discovering of Entities and Resources	[SWS_DM_01389] [SWS_DM_01404] [SWS_DM_01417] [SWS_DM_01420] [SWS_DM_01429] [SWS_DM_01442] [SWS_DM_01446] [SWS_DM_01449] [SWS_DM_01790] [SWS_DM_01797]
[RS_Diag_04259]	Fault Memory Access	[SWS_DM_01406] [SWS_DM_01407] [SWS_DM_01408] [SWS_DM_01409] [SWS_DM_01411] [SWS_DM_01412] [SWS_DM_01413] [SWS_DM_01414] [SWS_DM_01415] [SWS_DM_01416] [SWS_DM_01417] [SWS_DM_01418] [SWS_DM_01561]
[RS_Diag_04260]	Read / Write Access	[SWS_DM_01419] [SWS_DM_01420] [SWS_DM_01430] [SWS_DM_01431] [SWS_DM_01432] [SWS_DM_01433] [SWS_DM_01434] [SWS_DM_01435] [SWS_DM_01436] [SWS_DM_01437] [SWS_DM_01439] [SWS_DM_01440] [SWS_DM_01441] [SWS_DM_01442] [SWS_DM_01444] [SWS_DM_01445] [SWS_DM_01446] [SWS_DM_01447] [SWS_DM_01448] [SWS_DM_01449]
[RS_Diag_04261]	Configuration Access	[SWS_DM_01421] [SWS_DM_01422] [SWS_DM_01423] [SWS_DM_01424] [SWS_DM_01425] [SWS_DM_01426] [SWS_DM_01427] [SWS_DM_01428] [SWS_DM_01429] [SWS_DM_01787] [SWS_DM_01788] [SWS_DM_01860] [SWS_DM_01861] [SWS_DM_01862] [SWS_DM_01863] [SWS_DM_01864] [SWS_DM_01865] [SWS_DM_01866] [SWS_DM_01867] [SWS_DM_01868] [SWS_DM_01869] [SWS_DM_01870] [SWS_DM_01871] [SWS_DM_01872] [SWS_DM_01930]





Requirement	Description	Satisfied by
[RS_Diag_04262]	Control of Operations	[SWS_DM_01390] [SWS_DM_01391] [SWS_DM_01392] [SWS_DM_01393] [SWS_DM_01394] [SWS_DM_01395] [SWS_DM_01396] [SWS_DM_01397] [SWS_DM_01398] [SWS_DM_01399] [SWS_DM_01400] [SWS_DM_01401] [SWS_DM_01402] [SWS_DM_01403] [SWS_DM_01404] [SWS_DM_01405] [SWS_DM_01478] [SWS_DM_01479] [SWS_DM_01480] [SWS_DM_01481] [SWS_DM_01482] [SWS_DM_01491] [SWS_DM_01492] [SWS_DM_01493] [SWS_DM_01494] [SWS_DM_01495] [SWS_DM_01852]
[RS_Diag_04263]	Support of Target Modes	[SWS_DM_01379] [SWS_DM_01380] [SWS_DM_01381] [SWS_DM_01382] [SWS_DM_01383] [SWS_DM_01384] [SWS_DM_01385] [SWS_DM_01386] [SWS_DM_01387] [SWS_DM_01388] [SWS_DM_01389]
[RS_Diag_04264]	SOVD specific Locking/Semaphore mechanism	[SWS_DM_01375] [SWS_DM_01443] [SWS_DM_01473] [SWS_DM_01474] [SWS_DM_01475] [SWS_DM_01476] [SWS_DM_01477] [SWS_DM_01929]
[RS_Diag_04265]	Software Update	[SWS_DM_01796] [SWS_DM_01797] [SWS_DM_01798] [SWS_DM_01799] [SWS_DM_01800] [SWS_DM_01801] [SWS_DM_01802] [SWS_DM_01803] [SWS_DM_01804] [SWS_DM_01874] [SWS_DM_01875] [SWS_DM_01876] [SWS_DM_01877] [SWS_DM_01878] [SWS_DM_01879] [SWS_DM_01881] [SWS_DM_01882] [SWS_DM_01883] [SWS_DM_01884] [SWS_DM_01885] [SWS_DM_01886] [SWS_DM_01887] [SWS_DM_01888] [SWS_DM_01889] [SWS_DM_01890] [SWS_DM_01891] [SWS_DM_01893] [SWS_DM_01894] [SWS_DM_01895] [SWS_DM_01896] [SWS_DM_01898] [SWS_DM_01899] [SWS_DM_01900] [SWS_DM_01901] [SWS_DM_01902] [SWS_DM_01903] [SWS_DM_01904] [SWS_DM_01905] [SWS_DM_01906] [SWS_DM_01907] [SWS_DM_01908] [SWS_DM_01909] [SWS_DM_01910] [SWS_DM_01911] [SWS_DM_01912] [SWS_DM_01913] [SWS_DM_01914] [SWS_DM_01915] [SWS_DM_01916] [SWS_DM_01917] [SWS_DM_01918] [SWS_DM_01919] [SWS_DM_01920] [SWS_DM_01921] [SWS_DM_01922] [SWS_DM_01923] [SWS_DM_01924] [SWS_DM_01925]





Requirement	Description	Satisfied by
[RS_Diag_04266]	Bulk data	[SWS_DM_01789] [SWS_DM_01790] [SWS_DM_01791] [SWS_DM_01792] [SWS_DM_01793] [SWS_DM_01794] [SWS_DM_01795] [SWS_DM_01828] [SWS_DM_01829] [SWS_DM_01830] [SWS_DM_01831] [SWS_DM_01832] [SWS_DM_01833] [SWS_DM_01834] [SWS_DM_01835] [SWS_DM_01836] [SWS_DM_01837] [SWS_DM_01838] [SWS_DM_01839] [SWS_DM_01840] [SWS_DM_01841] [SWS_DM_01842] [SWS_DM_01843] [SWS_DM_01844] [SWS_DM_01845] [SWS_DM_01846] [SWS_DM_01847] [SWS_DM_01848] [SWS_DM_01849] [SWS_DM_01850] [SWS_DM_01851] [SWS_DM_01853] [SWS_DM_01854] [SWS_DM_01855] [SWS_DM_01856] [SWS_DM_01857] [SWS_DM_01858] [SWS_DM_01873] [SWS_DM_01926]
[RS_Diag_04267]	Logging	[SWS_DM_01928]
[RS_Diag_04268]	Authentication	[SWS_DM_01376] [SWS_DM_01469] [SWS_DM_01470] [SWS_DM_01471] [SWS_DM_01472] [SWS_DM_01483] [SWS_DM_01484] [SWS_DM_01485] [SWS_DM_01486] [SWS_DM_01487] [SWS_DM_01488] [SWS_DM_01489] [SWS_DM_01490] [SWS_DM_01781] [SWS_DM_01782] [SWS_DM_01783] [SWS_DM_01819] [SWS_DM_01820] [SWS_DM_01927]
[RS_Diag_04270]	Notify applications about the changes of enable conditions	[SWS_DM_01094] [SWS_DM_01740]
[RS_Ids_00810]	Basic SW security events	[SWS_DM_01760] [SWS_DM_01761] [SWS_DM_01762] [SWS_DM_01763] [SWS_DM_01764] [SWS_DM_01765] [SWS_DM_01766] [SWS_DM_01767] [SWS_DM_01768] [SWS_DM_01769] [SWS_DM_01770] [SWS_DM_01771] [SWS_DM_01772] [SWS_DM_01773] [SWS_DM_01774] [SWS_DM_01775] [SWS_DM_01776] [SWS_DM_01777] [SWS_DM_01778] [SWS_DM_01779] [SWS_DM_01780]
[RS_Main_01002]	AUTOSAR shall support service-oriented communication	[SWS_DM_00557] [SWS_DM_00595] [SWS_DM_00604]
[SRS_Eth_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_CONSTR_00206]
[SRS_Eth_00027]	DoIP diagnostic message shall have a format	[SWS_DM_00449]





Requirement	Description	Satisfied by
[SRS_Eth_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]
[SRS_Eth_00082]	DoIP shall implement a mechanism to retrieve Entity Status	[SWS_DM_00449]
[SRS_Eth_00083]	DoIP shall implement a mechanism to check if diagnostic testers are alive	[SWS_DM_00449]
[SRS_Eth_00084]	DoIP shall implement routing activation mechanism	[SWS_DM_00449]
[SWS_CORE_10001]	Definition of API function ara::core::Initialize	[SWS_DM_01571]
[SWS_CORE_10002]	Definition of API function ara::core::Deinitialize	[SWS_DM_01572] [SWS_DM_01573] [SWS_DM_01574]
[SWS_DM_01735]	MultipleCondition ConditionHandle Type Generation	[SWS_DM_01703]

**Table 6.1: RequirementsTracing**

## 6.1 Not applicable requirements

[SWS\_DM\_NA]{DRAFT} [These requirements are not applicable as they are not within the scope of this release.] ([RS\\_Diag\\_04059](#), [RS\\_Diag\\_04171](#), [RS\\_Diag\\_04195](#), [RS\\_Diag\\_04200](#), [RS\\_Diag\\_04218](#))

## 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 7.2 for details. The subcomponent of **DM** implementing a particular Transport Protocol is called a **Transport Protocol Handler**.

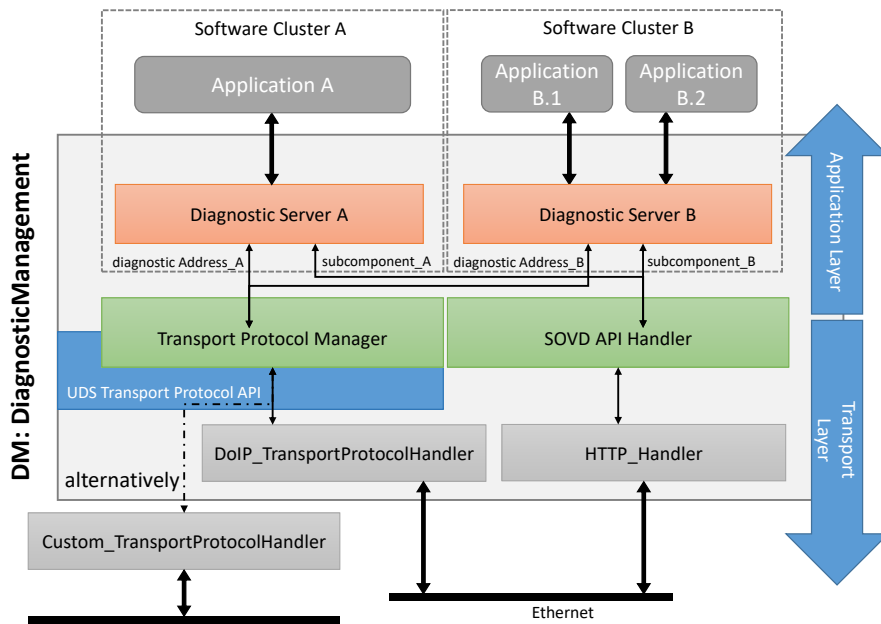
Besides UDS Transport Layer also communication via **SOVD** is supported according to [2]. The **SOVD** Transport Layer is covered in section 7.3 “**SOVD Transport Layer**”.

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**[2]. 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 7.4. In context of **SOVD** the diagnostic address corresponds to an **SOVD** subcomponent, see section 7.4 “**Diagnostic Server**”.

The link between the UDS Transport Layer and the Application Layer is implemented by the **Transport Protocol Manager** ( see subsection B.1.4 “**UdsTransportProtocolMgr Class**”), 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 subsection B.1.3 “**UdsTransportProtocolHandler Class**”) that handles the connection to the **Diagnostic Client**. Accordingly, the dispatching between **SOVD** Transport Layer and different **SOVD** subcomponents need to be handled.

A broad subcomponent view on **DM** is given as follows:





**Figure 7.1: Component view on Diagnostic Management**

## 7.1 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.

### 7.1.1 Startup

The startup procedure of the DM is treated as vendor-specific.

**[SWS\_DM\_01571]{DRAFT} Load persisted data** [On a DM startup trigger from Execution Management, DM shall restore all data, which was persisted from persistent memory.](SWS\_CORE\_10001)

### 7.1.2 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]{DRAFT} Graceful shutdown** [On a DM termination trigger from [Execution Management](#), DM shall gracefully shutdown itself.] ([SWS\\_CORE\\_10002](#))

**[SWS\_DM\_01573]{DRAFT} Stop all running Transport Protocol Handlers** [On a DM termination trigger from [Execution Management](#), DM shall call `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` on all started `UdsTransportProtocolHandlers`.] ([SWS\\_CORE\\_10002](#))

**[SWS\_DM\_01574]{DRAFT} Write data to be persisted** [Latest on a DM termination trigger from [Execution Management](#), DM shall write all data, which has the need of persistent storage, to persistent memory.] ([SWS\\_CORE\\_10002](#))

### 7.1.3 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 "7.4.4.1" the caching for DEM related features is described to be able to re-sync after a daemon restart.

**[SWS\_DM\_01580]{DRAFT}** [The DCM related parts shall behave like a session timeout according to [\[SWS\\_DM\\_00381\]](#) or priority handling in [\[SWS\\_DM\\_00427\]](#), [\[SWS\\_DM\\_00428\]](#), [\[SWS\\_DM\\_00429\]](#), and [\[SWS\\_DM\\_00430\]](#) in a conversation.] ()

## 7.2 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](#) [\[5\]](#). 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.

**[SWS\_DM\_00342]{DRAFT} Indication of UDS message reception** [UDS Transport Protocol implementation shall call `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` on its `ara::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`

](*RS\_Diag\_04168*)

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 `ara::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.

## 7.2.1 Support of proprietary UDS Transport Layer

The UDS Transport Protocol API is formally described in section B.1. This section describes the required interaction of the components using this API. Each (proprietary) UDS Transport Protocol implementation subclasses the abstract class `ara::diag::uds_transport::UdsTransportProtocolHandler`, which shall be provided by DM according to [*SWS\_DM\_00315*].

### 7.2.1.1 Initialization, Starting and Stopping of a proprietary UDS TransportLayer

[*SWS\_DM\_00329*]{DRAFT} **Lifecycle management of an UDS Transport Protocol implementation** [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 `ara::diag::uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler`.
- Initializing of the UDS Transport Protocol implementation by calling `ara::diag::uds_transport::UdsTransportProtocolHandler::Initialize`.
- Starting of the UDS Transport Protocol implementation by calling `ara::diag::uds_transport::UdsTransportProtocolHandler::Start`.

- Stopping of the UDS Transport Protocol implementation by calling `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop`.

](*RS\_Diag\_04168*)

**[SWS\_DM\_00330]{DRAFT} Construction of an UDS Transport Protocol implementation** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr` provided by DM.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00331]{DRAFT} Initialization of an UDS Transport Protocol implementation** [The DM shall call the `ara::diag::uds_transport::UdsTransportProtocolHandler::Initialize` during startup/initialization phase, before reporting `ApplicationState.kRunning` to the execution management.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01746]{DRAFT} Required successful initialization** [The DM shall call any method except of `GetPeriodicHandler` on `ara::diag::uds_transport::UdsTransportProtocolHandler` only after `ara::diag::uds_transport::UdsTransportProtocolHandler::Initialize` has returned `kInitializeOk`.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01745]{DRAFT} Behavior on failed transport protocol initialization** [If a `ara::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 `ara::diag::uds_transport::UdsTransportProtocolHandler`.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01744]{DRAFT} Processing of ChannelReestablished** [The DM shall process a call of `ara::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished` only after the call of `ara::diag::uds_transport::UdsTransportProtocolHandler::Start`.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01743]{DRAFT} Require a started UdsTransportProtocolHandler** [The DM shall call `ara::diag::uds_transport::UdsTransportProtocolHandler::Transmit` and `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` only after `ara::diag::uds_transport::UdsTransportProtocolHandler::Start` was called.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00332]{DRAFT} Starting of an UDS Transport Protocol implementation** [The DM shall call the `ara::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 `ara::diag::uds_transport::UdsTransportProtocolHandler::Initialize` has returned.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01742]{DRAFT} Asynchronous UDS protocol start** [The `ara::diag::uds_transport::UdsTransportProtocolHandler::Start` implementation shall be asynchronous and can continue starting the UDS protocol handler after the call has returned.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00333]{DRAFT} Stopping of an UDS Transport Protocol implementation** [The DM shall call the `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` method of each UDS Transport Protocol implementation it has started, if it is switching to state `ApplicationState.kTerminating`.] (*RS\_Diag\_04168*)

**[SWS\_DM\_01741]{DRAFT} No more method calls after stop** [After `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` has returned, the handler-plugin shall NOT call to `ara::diag::uds_transport::UdsTransportProtocolMgr` with any other method except of `ara::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped`.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00340]{DRAFT} Waiting for Stop confirmation** [After having called `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` method of any UDS Transport Protocol implementation, it shall wait for the corresponding `ara::diag::uds_transport::UdsTransportProtocolMgr::HandlerStopped` callback with the related `handlerId`, before it finally terminates the process.] (*RS\_Diag\_04168*)

### 7.2.1.2 UDS message reception on a proprietary UDS TransportLayer

**[SWS\_DM\_00347]{DRAFT} Channel identification in Indication** [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.] (*RS\_Diag\_04168*)

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

- The UDS `source address (SA)` in the clients/testers request which represent a technology/transport layer independent part.
- The technology/transport layer specific/dependent network endpoint source address, from which the request from the client originates. In Ethernet-based networks this typically is an IP-address/port number pair, while in CAN networks it is the CAN identifier of the CAN-TP message used by the client. In UDS on CAN (ISO ISO-15765-2[9]) 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]{DRAFT} Acceptance of UDS message reception** [If the *DM* is able to process the indicated request, `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` shall return a `std::pair` with `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` set to `kIndicationOk` and a `ara::diag::uds_transport::UdsMessagePtr`, which owns a valid `ara::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 `ara::diag::uds_transport::UdsMessage` object shall be one byte.]([RS\\_Diag\\_04168](#))

Note: For details about `std::pair` see [\[SWS\\_DM\\_00309\]](#).

**[SWS\_DM\_00392]{DRAFT} Properties of returned UdsMessage** [If the *DM* accepted the UDS message reception, the returned `ara::diag::uds_transport::UdsMessage` owned by `ara::diag::uds_transport::UdsMessagePtr` shall return a `ara::diag::uds_transport::ByteSpan` from `ara::diag::uds_transport::UdsMessage::GetPayload`, which shall be empty (i.e. `empty()` returns true, `size()` returns 0).]([RS\\_Diag\\_04168](#))

Note: In the normal case, where *DM* accepts the complete UDS request for processing, it will provide a `std::pair` with `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` set to `kIndicationOk` and a `ara::diag::uds_transport::UdsMessagePtr`, which owns a valid `ara::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 `ara::diag::uds_transport::UdsMessagePtr`, which owns a valid `ara::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 `ara::diag::uds_transport::UdsMessagePtr` with only enough capacity to be able to construct a valid negative response.

**[SWS\_DM\_00386]{DRAFT} Ignoring UDS message reception because DM is busy** [If the Transport Protocol Manager is calling `ara::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 `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` set to `kIndicationOccupied` and a `ara::diag::uds_transport::UdsMessagePtr` equal to `nullptr`.]([RS\\_Diag\\_04168](#))

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 `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage`. For handling of the functional `TesterPresent`, see chapter 7.4.2.8.21.

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]{DRAFT} Ignoring UDS message reception because DM has no (memory) resources** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` set to `kIndicationOverflow` and a `ara::diag::uds_transport::UdsMessagePtr` equal to `UdsMessagePtr(nullptr)`.] (*RS\_Diag\_04168*)

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]{DRAFT} Ignoring UDS message reception because of unknown target address** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` set to `kIndicationUnknownTargetAddress` and a `ara::diag::uds_transport::UdsMessagePtr` equal to `UdsMessagePtr(nullptr)`.] (*RS\_Diag\_04168*)

Note: For details about `std::pair` see [SWS\_DM\_00309].

**[SWS\_DM\_00388]{DRAFT} Filling provided UdsMessage** [If the DM returned `kIndicationOk` from the `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage`, the UDS Transport Protocol implementation shall fill the `ara::diag::uds_transport::UdsMessage` owned by `ara::diag::uds_transport::UdsMessagePtr` from the received UDS request starting from `SID` up to either `ara::diag::uds_transport::UdsMessage` full capacity or up to the entire received UDS request message, whatever happens first.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00345]{DRAFT} Forwarding of UDS message** [If the UDS Transport Protocol implementation has filled the payload of the returned `ara::diag::uds_transport::UdsMessagePtr`, it shall call `ara::diag::uds_transport::Ud-`

`sTransportProtocolMgr::HandleMessage` on its `ara::diag::uds_transport::UdsTransportProtocolMgr` reference ((see [SWS\_DM\_00330]) with the returned `ara::diag::uds_transport::UdsMessagePtr` as argument.) (RS\_Diag\_04168)

**[SWS\_DM\_00389]{DRAFT} Skipping Forwarding of UDS message** [If the DM returned a `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicationResult` NOT equal to `kIndicationOk` from the `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage`, the UDS Transport Protocol implementation shall NOT call `ara::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage`.] (RS\_Diag\_04168)

**[SWS\_DM\_00346]{DRAFT} Aborting of UDS message** [If the UDS Transport Protocol implementation has already called `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` (see [SWS\_DM\_00342]), but is not willing to call `ara::diag::uds_transport::UdsTransportProtocolMgr::HandleMessage` (maybe due to errors receiving the entire/remaining UDS request), it shall notify DM by calling `ara::diag::uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure` on its `ara::diag::uds_transport::UdsTransportProtocolMgr` reference ((see [SWS\_DM\_00330]) with the returned `ara::diag::uds_transport::UdsMessagePtr` as argument.) (RS\_Diag\_04168)

### 7.2.1.3 UDS message transmission on a proprietary UDS TransportLayer

**[SWS\_DM\_00348]{DRAFT} Transmission of UDS response message** [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 `ara::diag::uds_transport::UdsTransportProtocolHandler::Transmit` method of the UDS Transport Protocol implementation.] (RS\_Diag\_04168)

**[SWS\_DM\_00349]{DRAFT} Reuse channel identifier of Indication** [DM shall set the argument `channelId` in the `ara::diag::uds_transport::UdsTransportProtocolHandler::Transmit` call to the same value as in the Indication of the corresponding UDS request message (see [SWS\_DM\_00347]).] (RS\_Diag\_04168)

**[SWS\_DM\_00350]{DRAFT} Confirmation of UDS message transmission** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation` ((see [SWS\_DM\_00330]) setting the `message` argument to the `message` parameter of the `ara::diag::uds_transport::UdsTransportProtocolHandler::Transmit` call ([SWS\_DM\_00348]).] (RS\_Diag\_04168)



**[SWS\_DM\_00351]{DRAFT} Confirmation Result** [When the the network layer was able to send the UDS response message to the network, the `result` argument in the `ara::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation` shall be set to `kTransmitOk`, otherwise to `kTransmitFailed`.] (*RS\_Diag\_04168*)

#### 7.2.1.4 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 `ara::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 `ara::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]{DRAFT} Requesting Notification of a channel reestablishment** [The DM shall call the `ara::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.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00357]{DRAFT} Validity/lifetime of a Notification Request** [A notification request registered at a UDS Transport Protocol implementation according to [SWS\_DM\_00356] is valid only for the next call to `ara::diag::uds_transport::UdsTransportProtocolHandler::Start` until the following call to `ara::diag::uds_transport::UdsTransportProtocolHandler::Stop` of this UDS Transport Protocol implementation.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00358]{DRAFT} Notification of a channel reestablishment** [UDS Transport Protocol implementation shall call `ara::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.] (*RS\_Diag\_04168*)

**[SWS\_DM\_00359]{DRAFT} Persistent Storage of Notification Request** [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.] (*RS\_Diag\_04168*)

## 7.2.2 DoIP

**[SWS\_DM\_00475]{DRAFT} Support of DoIP based on ISO 13400-2** [The DM shall support DoIP ISO 13400-2[5] specification.] ([RS\\_Diag\\_04242](#))

Note: According to the ISO 13400-2[5] specification, the DoIP entity supports protocol version = 0xFF in the vehicle identification request message.

**[SWS\_DM\_00449]{DRAFT} Supported DoIP message types** [The DM shall support the DoIP message types listed in Table [\[SWS\\_DM\\_01568\]](#).] ([SRS\\_Eth\\_00026](#), [SRS\\_Eth\\_00080](#), [SRS\\_Eth\\_00082](#), [SRS\\_Eth\\_00083](#), [SRS\\_Eth\\_00084](#), [SRS\\_Eth\\_00027](#))

**[SWS\_DM\_01568]{DRAFT} supportedDoIpMessageTypes** [

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

**Table 7.1: Supported DoIp message types**

]()

**[SWS\_DM\_00855]{DRAFT} Providing the VIN in DoIP protocol messages** [DM shall retrieve the VIN required in some DoIP messages by reading the data from the DID with `DiagnosticDataIdentifier.representsVin` set to TRUE.] ([SRS\\_Eth\\_00026](#))

**[SWS\_DM\_00814]{DRAFT} Providing the PowerMode in DoIP protocol messages** [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`.] ([SRS\\_Eth\\_00026](#), [SRS\\_Eth\\_00080](#))

**[SWS\_DM\_01525]{DRAFT} `ara::diag::DoIPPowerMode` not yet offered when client requests DoIP PowerMode** [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::DoIPPower-`

`Mode` interface is not called yet by the Adaptive Application that provides it (as defined in the ISO 13400-2[5] specification).] ([SRS\\_Eth\\_00080](#))

**[SWS\_DM\_00813]{DRAFT} Providing the `GID` in DoIP protocol messages** [If the `DM` needs to know the `GID` and the status of the `GID` to be able to react or answer on any DoIP message, it shall obtain it by calling the method `ara::diag::DoIPGroupIdentification::GetGidStatus`.] ([SRS\\_Eth\\_00026](#))

**[SWS\_DM\_01527]{DRAFT} `ara::diag::DoIPGroupIdentification` not yet offered when `DM` needs to retrieve `GID`** [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[5] specification].] ([SRS\\_Eth\\_00026](#))

**[SWS\_DM\_00815]{DRAFT} When to send Vehicle announcement messages on interfaces without activation line control** [The `DM` gets notified, when to send out vehicle announcement messages on a network interface without activation line control (`isActivationLineDependent == FALSE`) by a call to method `ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement`, which `DM` has to provide. The method call contains the network interface identified via `networkInterfaceId` on which the announcement shall be sent.] ([RS\\_Diag\\_04242](#))

**[SWS\_DM\_00816]{DRAFT} Notification of activation line status change on activation line controlled network interfaces** [The `DM` gets notified, when the activation line status changes for activation line controlled network interfaces (`isActivationLineDependent == TRUE`) via software components providing an instance of `DiagnosticDoIPActivationLineInterface`. The `DM` shall identify for which network interface an instance of `DiagnosticDoIPActivationLineInterface` is providing the activation line status via call to method `ara::diag::DoIPActivationLine::GetNetworkInterfaceId`. Whenever the status of the activation line of the related network interface changes, the application calls `ara::diag::DoIPActivationLine::UpdateActivationLineState`.] ([RS\\_Diag\\_04242](#))

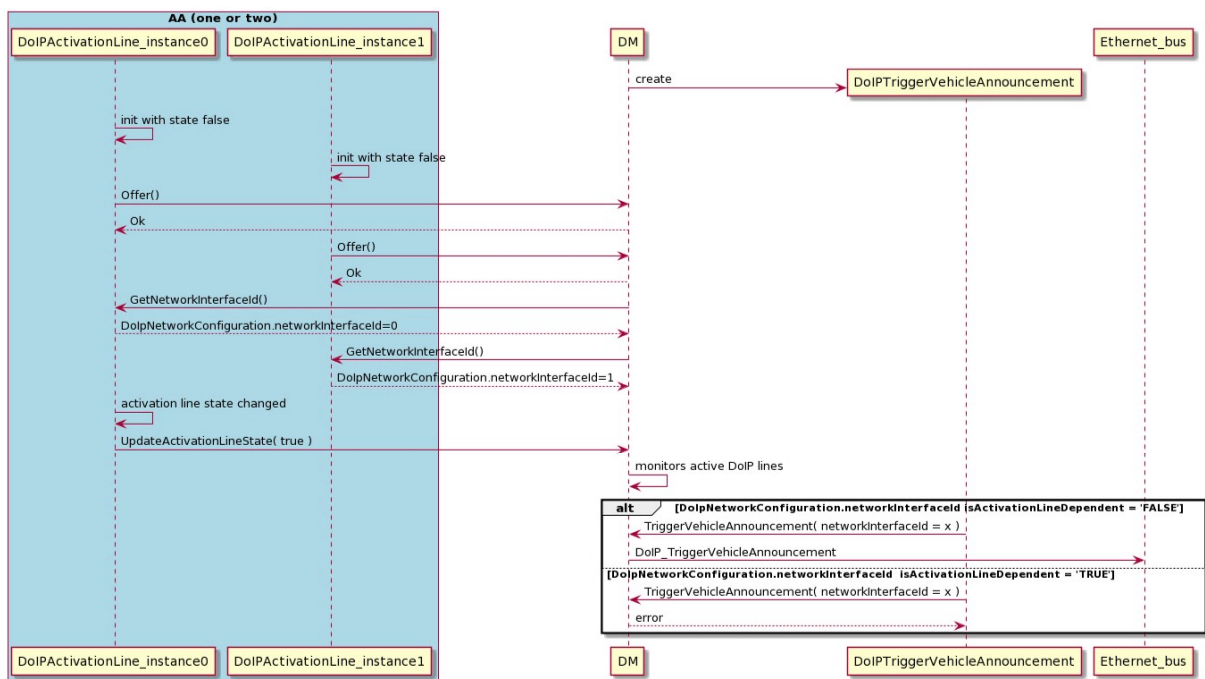
The `DM` needs to instantiate a singleton of `ara::diag::DoIPTriggerVehicleAnnouncement` only. The `ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement` method gets the according `networkInterfaceId` passed, to which the DoIP Vehicle Announcement Message (see [\[SWS\\_DM\\_00449\]](#)) shall be sent. Different `AAs` monitoring the DoIP Activation Line will inform the `DM` about an Activation Line toggle on its monitoring `networkInterfaceId`.

**[SWS\_DM\_01526]{DRAFT} `ara::diag::DoIPActivationLine` not yet offered on activation line controlled network interfaces** [As long as the Offer function of the `ara::diag::DoIPActivationLine` interface is not called by some Adaptive Application for a given activation line controlled network interface (`DoIpNetworkConfiguration.isActivationLineDependent == TRUE`) the DoIP communication shall

not be started on the related network interface, since the DoIP Activation Line state cannot be sensed by the DM (according to the ISO 13400-2[5] specification).]()

**[SWS\_DM\_01361]{DRAFT} Providing the EID in DoIP protocol messages** [If the DM needs to know the EID to be able to react or answer on any DoIP message, it shall obtain it by calling the method `ara::diag::DoIPEntityIdentification::GetEntityId` if configured according to [TPS\_MANI\_03294].] (*SRS\_Eth\_00026*)

**[SWS\_DM\_01528]{DRAFT} `ara::diag::DoIPEntityIdentification` not yet offered when DM needs to retrieve EID from Application** [If the DM shall send the EID while the Offer function of the `ara::diag::DoIPEntityIdentification` interface was not called yet by the Adaptive Application that provides it, the DM shall send a EID 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[5] specification.] (*SRS\_Eth\_00026*)



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

### 7.2.3 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]{DRAFT} Dispatching physical Request** [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](#).] ([RS\\_Diag\\_04216](#))

**[SWS\_DM\_00391]{DRAFT} Dispatching functional Request** [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](#).] ([RS\\_Diag\\_04216](#))

### 7.3 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]{DRAFT} DM as SOVD Server** [The [DM](#) shall act as [SOVD Server](#) as specified by [ASAM SOVD\[2\]](#) according to the configuration referenced by [SovdModuleInstantiation.communicationConnector](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01370]{DRAFT} DNS-Based Service Discovery and Multicast DNS for SOVD** [The [DM](#) shall support [DNS-SD](#) ([DNS-Based Service Discovery](#)) and [mDNS](#) ([Multicast DNS](#)) as specified by [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01371]{DRAFT} Secure Communication for SOVD using TLS** [The [DM](#) shall secure its [HTTP](#) connections using [TLS](#) according to the configuration referenced by [SovdModuleInstantiation.securePropsForTcp](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01372]{DRAFT} Representation of DM by SOVD component** [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](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01373]{DRAFT} Representation of Diagnostic Server instance by SOVD subcomponents** [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`.] (*RS\_Diag\_04256*)

**[SWS\_DM\_01374]{DRAFT} Dispatching of SOVD requests/responses** [Within the DM SOVD requests and responses shall be dispatched to and from the Diagnostic Server instances based on the SOVD path.] (*RS\_Diag\_04256*)

## 7.4 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 [10].

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]{DRAFT} Instantiation of Diagnostic Server** [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`.] (*RS\_Diag\_04216*)

Details on required configuration items are described in section 7.4.5.

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.

**[SWS\_DM\_01345]{DRAFT} Lifetime of MetalInfo** [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.] ([RS\\_Diag\\_04170](#))

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.

### 7.4.1 Interaction between DM and applications

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

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

**[SWS\_DM\_01529]{DRAFT} Behavior on failed `ara::diag` instantiation** [If a client application instantiates an `ara::diag` interface with an invalid instance specifier in the constructor, the DM shall treat it as a violation according to [[SWS\\_CORE\\_90006](#)] .] ([RS\\_AP\\_00149](#))

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

### 7.4.2 Diagnostic Communication Management

A central element in the handling of diagnostic communication is the term [Diagnostic Conversation](#), which is described in section [7.4.2.1](#). A UDS request is always processed in the context of a Diagnostic Conversation. A single Diagnostic Server can handle multiple Diagnostic Conversations in parallel.

#### 7.4.2.1 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]{DRAFT} Identification of a Diagnostic Client** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage`, see [SWS\_DM\_00347].] (*RS\_Diag\_04005*)

**[SWS\_DM\_00046]{DRAFT} Each Diagnostic Conversation has its own session resources** [The `Diagnostic Server instance` shall provide each `Diagnostic Conversation` with its own and independently managed diagnostic session, which can be any valid UDS session type.] (*RS\_Diag\_04119, RS\_Diag\_04006*)

**[SWS\_DM\_00047]{DRAFT} Each Diagnostic Conversation has its own security-level resources** [The `Diagnostic Server instance` shall provide each `Diagnostic Conversation` with its own and independently managed security-level.] (*RS\_Diag\_04005*)

**[SWS\_DM\_01259]{DRAFT} Validation of Security Level Locked in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`** [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`.] (*RS\_Diag\_04005*)

**[SWS\_DM\_01260]{DRAFT} Validation of Invalid Security Level in `ara::diag::Conversation::GetDiagnosticSecurityLevelShortName`** [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`.] (*RS\_Diag\_04005*)

**[SWS\_DM\_01261]{DRAFT} Validation of Invalid Session Level in `ara::diag::Conversation::GetDiagnosticSessionShortName`** [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`.] (*RS\_Diag\_04005*)

#### 7.4.2.1.1 Parallel Client Handling

The `DM` is capable of parallel processing of client requests, as long as all clients are in the default session. If one client switches to a non-default session, the `DM` will process only diagnostic requests from the conversation of that tester which has requested the



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

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

**Impact of SOVD** With the introduction of **SOVD**[2] 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]{DRAFT} Re-entrant ara::diag interface calls for service processing** [The **DM** shall only call a method on an interface for service processing in a re-entrant 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::ReentrancyType` and
- the value of this parameter is set to `kFully`.

](RS\_Diag\_04166)

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]{DRAFT} Re-entrant ara::diag interface calls for DID read processing** [The DM shall only call a read DID processing method on an interface for service processing in a re-entrant 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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` or `ara::diag::GenericDataIdentifier` and the same required port for different clients and
- the interface constructor has the parameter `ara::diag::DataIdentifier-ReentrancyType` and
- the value of read is set to `kFully`.

](RS\_Diag\_04166)

**[SWS\_DM\_00942]{DRAFT} Re-entrant ara::diag interface calls for DID write processing** [The DM shall only call a write DID processing method on an interface for service processing in a re-entrant 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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` or `ara::diag::GenericDataIdentifier` and the same required port for different clients and
- the interface constructor has the parameter `ara::diag::DataIdentifier-ReentrancyType` and
- the value of write is set to `kFully`.

](RS\_Diag\_04166)

**[SWS\_DM\_00943]{DRAFT} Re-entrant ara::diag interface calls for DID read and write processing** [The DM shall only call a write and read DID processing method on an interface for service processing in a re-entrant 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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` or `ara::diag::GenericDataIdentifier` and the same required port for different clients and
- the interface constructor has the parameter `ara::diag::DataIdentifier-ReentrancyType` and
- the value `readWrite` is set to `kFully`.

]([RS\\_Diag\\_04166](#))

**[SWS\_DM\_00944]{DRAFT} Validity of re-entrant ara::diag interface calls for DID processing** [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.]([RS\\_Diag\\_04166](#))

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.

**[SWS\_DM\_01375]{DRAFT} Behavior on locked SOVD** [If a service cannot be performed or a session change is not possible due to SOVD lock, NRC 0x21 (BusyRepeatRequest) shall be returned.]([RS\\_Diag\\_04264](#))

#### 7.4.2.1.2 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 [7.4.2.9](#)) or if **all** of the following conditions are satisfied:

- UDS request processing is finished by either
  - sending positive or final negative response and processing `ara::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.

#### 7.4.2.1.3 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]{DRAFT} Instantiation of Diagnostic Conversation Interface**  
[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 `maxTesterConnections`.] (*RS\_Diag\_04166*)

**[SWS\_DM\_00841]{DRAFT} Assignment of Diagnostic Conversation to Service Instances**  
[On establishment of a new *Diagnostic Conversation*, the *Diagnostic Server instance* shall assign this *Diagnostic Conversation* to an inactive `ara::diag::Conversation` class Instance, i.e. the field value of `ara::diag::ActivityStatusType` is set to `kInactive`. After assignment, the fields of the `ara::diag::Conversation` class Instance shall be updated according to the state of the given *Diagnostic Conversation*, i.e.,

- `ara::diag::ActivityStatusType` set to `kActive`,
- `ara::diag::Conversation::ConversationIdentifierType` matching the values of `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` call, that initiated the creation of this *Diagnostic Conversation* (see [SWS\_DM\_00347]),
- a call to `ara::diag::Conversation::GetDiagnosticSession` will return the *Diagnostic Session* of this *Diagnostic Conversation*,
- a call to `ara::diag::Conversation::GetDiagnosticSecurityLevel` will return the *Diagnostic Security Level* of this *Diagnostic Conversation*.

] (*RS\_Diag\_04166*)

**[SWS\_DM\_00844]{DRAFT} Updating DiagnosticConversation Service Instance fields**  
[During the life-cycle of a *Diagnostic Conversation*, the *Diagnostic Server instance* shall update the fields of the assigned `ara::diag::Conversation` class instance according to any change of the state of the *Diagnostic Conversation*.] (*RS\_Diag\_04166*)

**[SWS\_DM\_00843]{DRAFT} Reset Service Instance fields on end of Diagnostic Conversation** [If the life-cycle of a [Diagnostic Conversation](#) ends, the [Diagnostic Server instance](#) shall reset

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

of the assigned `ara::diag::Conversation` class instance.]([RS\\_Diag\\_04166](#))

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]{DRAFT} Default session change trigger from AAs** [If `ara::diag::Conversation::ResetToDefaultSession` method is called, the [Diagnostic Server instance](#) shall complete the latest ongoing request and then switch the [Diagnostic Session](#) of this [Diagnostic Conversation](#) to [Default Session](#).]([RS\\_Diag\\_04006](#))

**[SWS\_DM\_01355]{DRAFT} Consecutive registration of notifier with SetActivityNotifier()** [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.]([RS\\_Diag\\_04169](#))

**[SWS\_DM\_01356]{DRAFT} Consecutive registration of notifier with SetDiagnosticSessionNotifier()** [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.]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04208](#))

**[SWS\_DM\_01357]{DRAFT} Consecutive registration of notifier with SetSecurityLevelNotifier()** [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.]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04208](#))

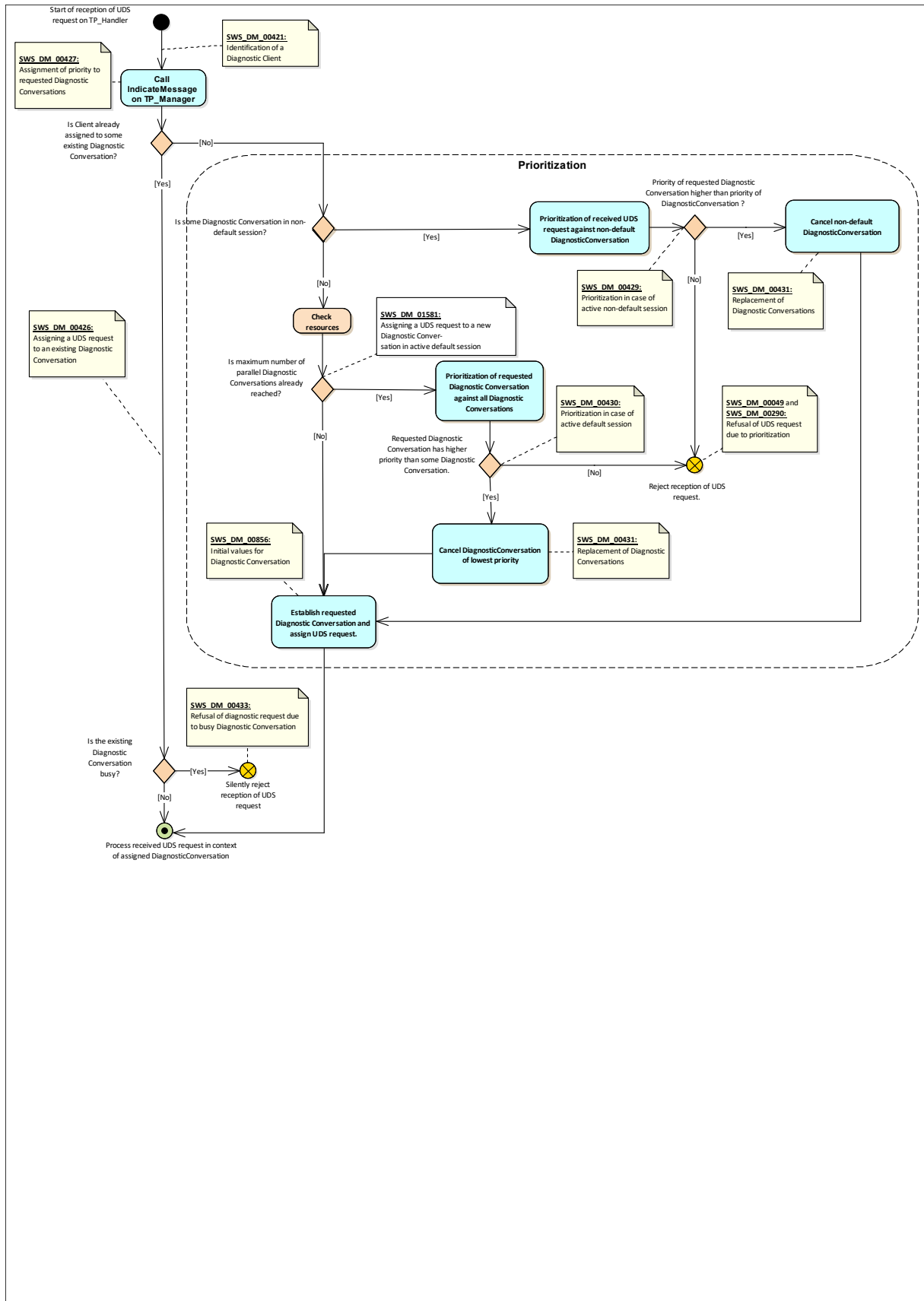
#### 7.4.2.2 Assignment of UDS requests to Diagnostic Conversations

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]{DRAFT} Assigning a UDS request to an existing Diagnostic Conversation** [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*.] (*RS\_Diag\_04166*)

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 7.4.2.1.2).

**[SWS\_DM\_01581]{DRAFT} Assigning a UDS request to a new Diagnostic Conversation in active default session** [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.] (*RS\_Diag\_04166*)

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].

#### 7.4.2.2.1 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\_00427]{DRAFT} Priority of a Diagnostic Conversation** [The *Diagnostic Server instance* shall take as the priority of a *Diagnostic Conversation* the respective value provided by `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage` that established the *Diagnostic Conversation*.] (*RS\_Diag\_04166*)

**[SWS\_DM\_00428]{DRAFT} Treatment of priority values** [The *Diagnostic Server instance* shall consider a lower value as higher priority and vice versa. In particular, priority value 0 represents highest priority.] (*RS\_Diag\_04166*)

**[SWS\_DM\_00429]{DRAFT} Prioritization in active non-default session** [If a *Diagnostic Conversation* is in non-default session, the *Diagnostic Server* shall compare the priority of the requested *Diagnostic Conversation* against the priority of the given *Diagnostic Conversation* in non-default session. If the priority of the re-



requested *Diagnostic Conversation* is higher than the priority of the *Diagnostic Conversation* in non-default *Session*, the *Diagnostic Server instance* shall replace the *Diagnostic Conversation* in non- default session by the requested *Diagnostic Conversation* according to [SWS\_DM\_00431] and assign the UDS request to the newly established *Diagnostic Conversation*.](RS\_Diag\_04166)

**[SWS\_DM\_00430]{DRAFT} Prioritization against all Diagnostic Conversations in active default session** [On prioritization, the *Diagnostic Server instance* shall compare the priority of the requested *Diagnostic Conversation* against the priorities of the existing *Diagnostic Conversations*:

- If all priorities of the existing *Diagnostic Conversations* are higher or equal to the priority of the requested *Diagnostic Conversation*, the *Diagnostic Server instance* shall refuse the UDS request according to [SWS\_DM\_00049] and [SWS\_DM\_00290].
- If some priority of the existing *Diagnostic Conversations* is lower than the priority of the requested *Diagnostic Conversation*, the *Diagnostic Server instance* shall replace the *Diagnostic Conversation* of lowest priority by the requested *Diagnostic Conversation* according to [SWS\_DM\_00431] and assign the UDS request to the newly established *Diagnostic Conversation*.

](RS\_Diag\_04166)

#### 7.4.2.2.2 Replacement of Diagnostic Conversations and initial values

**[SWS\_DM\_00431]{DRAFT} Replacement of Diagnostic Conversations** [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.](RS\_Diag\_04167)

**[SWS\_DM\_00856]{DRAFT} Initial values for Diagnostic Conversation** [For a newly established *Diagnostic Conversation*, the *Diagnostic Server instance* shall use the following initial values:

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

](RS\_Diag\_04166)

### 7.4.2.2.3 Refusal of incoming diagnostic request

**[SWS\_DM\_00433]{DRAFT} Refusal of diagnostic request due to busy Diagnostic Conversation** [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 [\[SWS\\_DM\\_00386\]](#).] ([RS\\_Diag\\_04020](#))

**[SWS\_DM\_00049]{DRAFT} Refusal of diagnostic request due to prioritization with BusyRepeatRequest** [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.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00290]{DRAFT} Refusal of diagnostic request due to prioritization without response** [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.] ([RS\\_Diag\\_04167](#))

### 7.4.2.3 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.

#### 7.4.2.3.1 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]{DRAFT} Get ClientAuthentication Instance** [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.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01203]{DRAFT} GetAll ClientAuthentication Instance** [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.]([RS\\_Diag\\_04240](#))

#### 7.4.2.3.2 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]{DRAFT} Support for authentication per Diagnostic Client** [The `Diagnostic Server instance` shall support the Authentication service independently for every `Diagnostic Client`.]([RS\\_Diag\\_04251](#))

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

**[SWS\_DM\_01204]{DRAFT} Default Authentication Role** [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`.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01205]{DRAFT} Default Authentication State** [On startup, the default Authentication state for a client shall be ‘`kDeAuthenticated`’].([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01206]{DRAFT} Set AuthenticationRole** [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.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01570]{DRAFT} Lifetime of overridden default roles** [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]{DRAFT} Get Authentication State** [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.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01208]{DRAFT} Authentication State Change Notifier** [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.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01209]{DRAFT} Temporarily change Default Roles** [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.]([RS\\_Diag\\_04239](#))

**[SWS\_DM\_01210]{DRAFT} DeAuthenticate due to client inactivity** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation` from this client no further request was received from the same client for a time of `authenticationTimeout`.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01211]{DRAFT} Transition to DeAuthenticated state on S3server timeout** [On an S3 Server timeout, the `Diagnostic Server instance` shall reset the Authentication State to `kDeAuthenticated` for the client which timed out.]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01212]{DRAFT} Transition from Authenticated to DeAuthenticated State** [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

]([RS\\_Diag\\_04240](#))

**[SWS\_DM\_01360]{DRAFT} Consecutive registration of notifier with ClientAuthentication::SetNotifier()** [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.] (*RS\_Diag\_04251*)

#### 7.4.2.3.3 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]{DRAFT} Set DynamicAccessList** [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.] (*RS\_Diag\_04240*)

**[SWS\_DM\_01215]{DRAFT} Extend the DynamicAccessList** [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.] (*RS\_Diag\_04240*)

**[SWS\_DM\_01216]{DRAFT} Revoke an authentication** [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’.] (*RS\_Diag\_04240*)

**[SWS\_DM\_01217]{DRAFT} Refresh timeouts** [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].

](RS\_Diag\_04240)

#### 7.4.2.3.4 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.

**[SWS\_DM\_01214]{DRAFT} Default DynamicAccessList** [On startup, the `DynamicAccessList` of all clients shall be empty.] (RS\_Diag\_04240)

**[SWS\_DM\_01218]{DRAFT} Building a new DynamicAccessList** [If any of the overloads of the method `ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder` of the class `ara::diag::DiagnosticServiceDynamicAccessList` is called by the application, the `Diagnostic Server instance` shall create a new `DynamicAccessList` beginning with the single-byte or byte-string pattern passed by the application. The `Diagnostic Server instance` shall return an instance of the class `ara::diag::DynamicAccessListDiagServiceBuilder` to the application.] (RS\_Diag\_04240)

**[SWS\_DM\_01219]{DRAFT} Adding patterns to a DynamicAccessList** [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.] (RS\_Diag\_04240)

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

**[SWS\_DM\_01220]{DRAFT} Adding wildcards to a DynamicAccessList** [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.] (RS\_Diag\_04240)

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

**[SWS\_DM\_01221]{DRAFT} End patterns of a DynamicAccessList** [If any of the overloads of the method `ara::diag::DynamicAccessListDiagService-`

`Builder::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`.] (*RS\_Diag\_04240*)

**[SWS\_DM\_01222]{DRAFT} Finalize a DynamicAccessList** [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`.] (*RS\_Diag\_04240*)

After successful execution of this method, the application may use the created `DynamicAccessList` as described in [SWS\_DM\_01213].

#### 7.4.2.4 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]{DRAFT} Authentication disabled** [If `DiagnosticAccessPermission.authenticationEnabled` does not exist, no authentication checks shall be performed.] ()

**[SWS\_DM\_01223]{DRAFT} Diagnostic service role verification** [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
- 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`

](RS\_Diag\_04233)

**[SWS\_DM\_01224]{DRAFT} Diagnostic service dynamic access-rights verification**  
 [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.]  
 (RS\_Diag\_04233)

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]{DRAFT} Response behavior of services without access rights**  
 [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.] (RS\_Diag\_04233)

**[SWS\_DM\_01772]{DRAFT} Reporting 'authentication required' to IdsM**  
 [In case a diagnostic service request is not having the required authentication, the *DM* shall report a security event SEV\_UDS\_REQUESTED\_ACTIONS\_REQUIRES\_AUTHENTICATION to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01931].]  
 (RS\_IdS\_00810)

**[SWS\_DM\_01931]{DRAFT}** [

Context Data:	SID and subfunction, if available, of the requested diagnostic service
---------------	--

**Table 7.2: SEV on 'authentication required'**

]()

**[SWS\_DM\_01376]{DRAFT} Response behavior of SOVD services without access rights**  
 [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.] (RS\_Diag\_04268)



### 7.4.2.5 UDS request Validation/Verification

**[SWS\_DM\_00096]{DRAFT} Validation Steps and Order** [The *Diagnostic Server instance* shall execute the request validation, negative response code determination and processing according to ISO 14229-1[1].] (*RS\_Diag\_04196, RS\_Diag\_04203*)

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 7.4.2.5.4
- **SID supported?** Decision according to section 7.4.2.5.2
- **SID supported in active session?** Decision according to section 7.4.2.5.3
- **SID security check o.k.?** Decision according to section 7.4.2.5.3
- **supplier-specific failure detected?** Decision by applying supplier-specific checks according to section 7.4.2.5.4

**[SWS\_DM\_00097]{DRAFT} Abort on failed verification step** [Whenever one of the verification steps fails, further processing of the request shall be aborted and a negative response shall be sent back.] (*RS\_Diag\_04196*)

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

#### 7.4.2.5.1 UDS request format checks

**[SWS\_DM\_00098]{DRAFT} UDS message checks** [The *Diagnostic Server instance* shall check, whether the diagnostic request is syntactically correct. I.e. whether it conforms to ISO 14229-1 message format specification. If it does not conform, the Verification shall be considered as failed and the negative response code shall be 0x13 (*incorrectMessageLengthOrInvalidFormat*).] (*RS\_Diag\_04203*)

**[SWS\_DM\_01778]{DRAFT} Reporting 'incorrect message length or invalid format' to IdsM** [In case a diagnostic service request is not correct in the length or the format, the *DM* shall report a security event SEV\_UDS\_INCORRECT\_MESSAGE\_LENGTH\_OR\_FORMAT to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01932].] (*RS\_IdS\_00810*)

**[SWS\_DM\_01932]{DRAFT}** [

Context Data:	SID and subfunction, if available, of the requested diagnostic service
---------------	--

**Table 7.3: SEV on 'Incorrect request message length or format'**

]0

#### 7.4.2.5.2 Supported service checks

**[SWS\_DM\_00099]{DRAFT} Supported Service SID level checks** [The *Diagnostic Server instance* shall check, whether there is a configured internal or external service processor for the incoming diagnostic request. If there is no service processor on *SID* level, the Verification shall be considered as failed and the negative response code shall be 0x11 (*serviceNotSupported*).] (*RS\_Diag\_04203*)

**[SWS\_DM\_01780]{DRAFT} Reporting service not supported to IdsM** [shall report a security event SEV\_UDS\_SERVICE\_NOT\_SUPPORTED to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01933].] (*RS\_Ids\_00810*)

**[SWS\_DM\_01933]{DRAFT}** [

Context Data:	SID of the requested diagnostic service
---------------	---

**Table 7.4: SEV on 'Service not supported'**

]0

**[SWS\_DM\_00100]{DRAFT} Supported Service subfunction level checks** [The *Diagnostic Server instance* shall check, whether there is a configured internal or external service processor for the incoming diagnostic request. If there exists a service processor on *SID* level, but not for the subfunction of the request, the Verification shall be considered as failed and the negative response code shall be 0x12 (*subFunctionNotSupported*).] (*RS\_Diag\_04203*)

**[SWS\_DM\_01779]{DRAFT} Reporting 'service subfunction not supported' to IdsM** [In case a diagnostic service subfunction is not supported, the *DM* shall report a security event SEV\_UDS\_SUBFUNCTION\_NOT\_SUPPORTED to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01934].] (*RS\_Ids\_00810*)

**[SWS\_DM\_01934]{DRAFT}** [

Context Data:	SID and subfunction of the requested diagnostic service
---------------	---

**Table 7.5: SEV on 'Subfunction not supported'**

]0

#### 7.4.2.5.3 Session and Security Checks

**[SWS\_DM\_00101]{DRAFT} Session Access SID level Permission** [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 *DiagnosticAc-*

`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,

the Verification shall be considered as failed and the negative response code shall be 0x7F (`serviceNotSupportedInActiveSession`).] ([RS\\_Diag\\_04203](#), [RS\\_Diag\\_04006](#))

**[SWS\_DM\_00102]{DRAFT} Session Access subfunction level Permission** [The `Diagnostic Server instance` shall check, whether the service processor (`DiagnosticServiceInstance`), which is assigned to handle the service has the permission to process the service in the current Diagnostic Session according to its `DiagnosticAccessPermission.diagnosticSession`. If `DiagnosticServiceInstance` has no access permissions in the current Diagnostic Session and:

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

the Verification shall be considered as failed and the negative response code shall be 0x7E (`subFunctionNotSupportedInActiveSession`).] ([RS\\_Diag\\_04203](#), [RS\\_Diag\\_04006](#))

**[SWS\_DM\_00103]{DRAFT} Security Access level Permission** [The `Diagnostic Server instance` shall check, whether the service processor (`DiagnosticServiceInstance`), which is assigned to handle the service has the permission to process the service in the current Security-Level according to its `DiagnosticAccessPermission.securityLevel`. If `DiagnosticServiceInstance` has no access permissions in the current Security-Level, the Verification shall be considered as failed and the negative response code shall be 0x33 (`securityAccessDenied`).] ([RS\\_Diag\\_04203](#), [RS\\_Diag\\_04005](#))

**[SWS\_DM\_01777]{DRAFT} Reporting 'security access denied' to IdsM** [In case a diagnostic service request is not allowed in the current security level, the `DM` shall report a security event `SEV_UDS_SECURITY_ACCESS_DENIED` to `IdsM` (see [11] and table [\[SWS\\_DM\\_01760\]](#)) with the context data given in the table x4.] ([RS\\_Ids\\_00810](#))

Context Data:	SID and subfunction, if available, of the requested diagnostic service
---------------	--

**Table 7.6: SEV on 'security access denied'**

**[SWS\_DM\_00450]{DRAFT} Security Access subfunction level Permission** [The `Diagnostic Server instance` shall check, whether the service processor (`DiagnosticServiceInstance`), which is assigned to handle the service has the per-

mission to process the service in the current Security Level according to its `DiagnosticAccessPermission.securityLevel`. If `DiagnosticServiceInstance` has no access permissions in the current Security Level and:

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

the Verification shall be considered as failed and the negative response code shall be `0x33 (securityAccessDenied)`.] (*RS\_Diag\_04203*)

#### 7.4.2.5.4 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]{DRAFT} Support of manufacturer service validations** [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.] (*RS\_Diag\_04199*)

**[SWS\_DM\_01253]{DRAFT} Support of supplier service validations** [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.] (*RS\_Diag\_04199*)

**[SWS\_DM\_01254]{DRAFT} Continue service processing after validation** [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].] (*RS\_Diag\_04199*)

**[SWS\_DM\_01255]{DRAFT} NRC after failed service validation** [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 [SWS\_DM\_00547] shall be sent as response.] (*RS\_Diag\_04199*)

**[SWS\_DM\_00859]{DRAFT} Confirmation of service processing** [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 `ara::diag::uds_transport::UdsTransportProtocolMgr::TransmitConfirmation`. In any other case, it shall be the last step of request processing.] (*RS\_Diag\_04019, RS\_Diag\_04172*)

**[SWS\_DM\_00860]{DRAFT} No service processing** [If any `ara::diag::ServiceValidation::Validate` returns `kNoProcessingNoResponse`, the `Diagnostic Server instance` shall discard the message of received diagnostic request.] (*RS\_Diag\_04196, RS\_Diag\_04199*)

#### 7.4.2.5.5 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]{DRAFT} Configurable environment condition checks** [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.] (*RS\_Diag\_04199*)

**[SWS\_DM\_00112]{DRAFT} Condition check definition** [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`.] (*RS\_Diag\_04199*)

**[SWS\_DM\_00286]{DRAFT} Configurable environmental condition check execution** [The `Diagnostic Server instance` shall execute an environmental condition check before executing the requested service if defined. (see `DiagnosticEnvironmentalCondition` element from DEXT [3]).] (*RS\_Diag\_04199*)

**[SWS\_DM\_00287]{DRAFT} Configurable environmental condition check criteria** [The environmental condition check shall be done by evaluation of the configured `DiagnosticEnvConditionFormula`.] (*RS\_Diag\_04199*)

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

**[SWS\_DM\_00288]{DRAFT} Configurable environmental condition check evaluates to TRUE** [If the computation of the `DiagnosticEnvConditionFormula` evaluated to `TRUE`, the `Diagnostic Server instance` shall execute the requested service.](*RS\_Diag\_04199*)

**[SWS\_DM\_00970]{DRAFT} Behavior of failed data element retrieval** [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.](*RS\_Diag\_04199*)

**[SWS\_DM\_00289]{DRAFT} Configurable environmental condition check evaluates to FALSE** [The `Diagnostic Server instance` shall send the `NRC` defined in `nrcValue`, if the computation of the `DiagnosticEnvConditionFormula` evaluated to `FALSE`. If `nrcValue` does not define a `NRC`, the `Diagnostic Server instance` shall send `NRC 0x22 (ConditionsNotCorrect)`.](*RS\_Diag\_04199*)

#### 7.4.2.6 UDS response handling

**[SWS\_DM\_01258]{DRAFT} Response handling** [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.](*RS\_Diag\_04196, RS\_Diag\_04020*)

**[SWS\_DM\_00368]{DRAFT} DM takes care of Response Pending Messages** [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].](*RS\_Diag\_04016*)

**[SWS\_DM\_00369]{DRAFT} Maximum number of busy responses** [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)`.](*RS\_Diag\_04016*)

**[SWS\_DM\_01257]{DRAFT} ResourceTemporarilyNotAvailable NRC handling** [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)`.]()

#### 7.4.2.7 Keep track of active non-default sessions

**[SWS\_DM\_00380]{DRAFT} Support for S3 timer** [The [Diagnostic Server instance](#) shall provide support for  $S3_{Server}$  (session timeout) with a server specific value according to [\[SWS\\_DM\\_01747\]](#). The timer handling shall be implemented according to ISO 14229-2[12].] ([RS\\_Diag\\_04006](#))

**[SWS\_DM\_01747]{DRAFT} S3 timer value** [If the parameter [DiagnosticSessionControlClass.s3ServerTimeout](#) is not configured, the [Diagnostic Server instance](#) (Software Cluster), shall take the value of 5000ms as the  $S3_{Server}$  timeout. If the parameter [DiagnosticSessionControlClass.s3ServerTimeout](#) is configured, the [Diagnostic Server Instance](#) (Software Cluster) shall take the configured value as the  $S3_{Server}$  timeout.] ()

**[SWS\_DM\_00381]{DRAFT} Session timeout** [Whenever a non-default session is active and when the session timeout ( $S3_{Server}$ ) is reached without receiving any diagnostic request, the [Diagnostic Server instance](#) shall reset to the default session state. [Diagnostic Server instance](#) internal states for service processing shall be reset according to ISO 14229-2[12].] ([RS\\_Diag\\_04006](#))

**[SWS\_DM\_00382]{DRAFT} Session timeout start** [The session timeout timer ( $S3_{server}$ ) shall be started on

- Completion of any final response message or an error indication during sending of the response ([\[SWS\\_DM\\_00312\]](#))
- Completion of the requested action in case no response message (positive and negative) is required / allowed.
- In case of an error during the reception of a multi-frame request message ([\[SWS\\_DM\\_00310\]](#))

Start of  $S3_{Server}$  means reset the timer and start counting from the beginning.] ([RS\\_Diag\\_04006](#))

**[SWS\_DM\_00383]{DRAFT} Session timeout stop** [The session timeout timer ( $S3_{Server}$ ) shall be stopped when the reception of an [UDS](#) message was indicated ([\[SWS\\_DM\\_00309\]](#)).] ([RS\\_Diag\\_04006](#))

**[SWS\_DM\_00812]{DRAFT} Re-enabling on transition to default session** [If [DTC](#) setting is disabled and [DM](#) is transitioning into default session, then [DM](#) shall enable the [DTC](#) setting again.] ([RS\\_Diag\\_04006](#))

#### 7.4.2.8 UDS service processing

This chapter describes the [UDS](#) service processing behavior of the [Diagnostic Server instance](#).

**[SWS\_DM\_00127]{DRAFT} Availability of diagnostic service processors** [The [Diagnostic Server instance](#) shall provide a service processor on SID level for all

services by existence of a `DiagnosticServiceClass` referenced by a `DiagnosticServiceInstance.serviceClass`.] ([RS\\_Diag\\_04196](#))

#### 7.4.2.8.1 Supported UDS Services

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

SID	Service	Support Type	Reference
0x10	DiagnosticSessionControl	Internally	<a href="#">7.4.2.8.4</a>
0x11	ECUReset	Externally	<a href="#">7.4.2.8.5</a>
0x14	ClearDiagnosticInformation	Internally	<a href="#">7.4.2.8.6</a>
0x19	ReadDTCInformation	Internally	<a href="#">7.4.2.8.7</a>
0x22	ReadDataByIdentifier	Internally & Externally	<a href="#">7.4.2.8.8</a>
0x27	SecurityAccess	Internally & Externally	<a href="#">7.4.2.8.9</a>
0x28	CommunicationControl	Externally	<a href="#">7.4.2.8.10</a>
0x29	Authentication	Externally	<a href="#">7.4.2.8.11</a>
0x2A	ReadDataByPeriodicIdentifier	Internally	<a href="#">7.4.2.8.12</a>
0x2C	DynamicallyDefineDataIdentifier	Internally	<a href="#">7.4.2.8.13</a>
0x2E	WriteDataByIdentifier	Externally	<a href="#">7.4.2.8.14</a>
0x31	RoutineControl	Externally	<a href="#">7.4.2.8.15</a>
0x34	RequestDownload	Externally	<a href="#">7.4.2.8.16</a>
0x35	RequestUpload	Externally	<a href="#">7.4.2.8.17</a>
0x36	TransferData	Externally	<a href="#">7.4.2.8.18</a>
0x37	RequestTransferExit	Externally	<a href="#">7.4.2.8.19</a>
0x38	RequestFile Transfer	Externally	<a href="#">7.4.2.8.20</a>
0x3E	TesterPresent	Internally	<a href="#">7.4.2.8.21</a>
0x85	ControlDTCSetting	Internally	<a href="#">7.4.2.8.22</a>
0x86	ResponseOnEvent	Internally	<a href="#">7.4.2.8.23</a>

**Table 7.7: 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.

#### 7.4.2.8.2 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]{DRAFT} Supported addressAndLengthFormatIdentifier** [The Diagnostic Server instance shall support for each nibble of the addressAndLengthFormatIdentifier a value between 1 and 8.] (RS\_Diag\_04120)

**[SWS\_DM\_00130]{DRAFT} Not supported addressAndLengthFormatIdentifier** [The Diagnostic Server instance shall send the negative response 0x31 (requestOutOfRange), if an addressAndLengthFormatIdentifier with a value outside the range between 1 and 8 is received.] (RS\_Diag\_04120)

**[SWS\_DM\_01762]{DRAFT} Reporting 'Sequence error' to IdsM** [In case a diagnostic service is received, which results in a negative response with NRC 0x24 (requestSequenceError), the DM shall report a security event SEV\_UDS\_REQUEST\_SEQUENCE\_ERROR (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01935].] (RS\_Ids\_00810)

**[SWS\_DM\_01935]{DRAFT}** [

Context Data:	SID and subfunction, if available, of the requested diagnostic service
---------------	--

**Table 7.8: SEV on 'Sequence error'**

]()

**[SWS\_DM\_01761]{DRAFT} Reporting 'Request out of range' to IdsM** [In case a diagnostic service is received, which results in a negative response with NRC 0x31 (requestOutOfRange), the DM shall report a security event SEV\_UDS\_REQUEST\_OUT\_OF\_RANGE (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01936].] (RS\_Ids\_00810)

**[SWS\_DM\_01936]{DRAFT}** [

Context Data:	SID and subfunction, if available, of the requested diagnostic service
---------------	--

**Table 7.9: SEV on 'Request out of range'**

]()

### 7.4.2.8.3 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]{DRAFT} DiagnosticDataElement serialization respecting the data element endianness for typed interfaces** [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`.]()

Note: Typed interfaces are referenced in chapter 8.3.2 “C++ Diagnostic generated API Interfaces”.

#### 7.4.2.8.4 Service 0x10 – DiagnosticSessionControl

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

**[SWS\_DM\_00226]{DRAFT} Support of UDS service DiagnosticSessionControl** [The `Diagnostic Server instance` shall provide the `UDS` service 0x10 `DiagnosticSessionControl` according to ISO 14229-1[1].] (*RS\_Diag\_04198*)

**[SWS\_DM\_00227]{DRAFT} Check for supported sessions** [If the Subfunction addressed by the `DiagnosticSessionControl` according to [SWS\_DM\_00226] is not supported by the configuration, i.e., there is no `DiagnosticSession` configured with `id` matching the requested Subfunction value, the `Diagnostic Server instance` shall return a `NRC 0x12 (SubFunctionNotSupported)`.] (*RS\_Diag\_04196*)

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

**[SWS\_DM\_00228]{DRAFT} Switch to requested Diagnostic Session** [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`.] (*RS\_Diag\_04198*)

**[SWS\_DM\_00845]{DRAFT} Notification about session change** [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.] (*RS\_Diag\_04208*)

#### 7.4.2.8.5 Service 0x11 – ECUReset

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

**[SWS\_DM\_00234]{DRAFT} Support of UDS service ECUReset** [The `DiagnosticServer` instance shall provide the UDS service 0x11 ECU Reset according to ISO 14229-1[1].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00235]{DRAFT} ECUReset service processing** [The `DiagnosticServer` instance shall first call the method `ara::diag::EcuResetRequest::RequestReset` of the `ara::diag::EcuResetRequest` class instance to process an ECU Reset except for Enable-/DisableRapidPowerShutdown services.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01018]{DRAFT} ECUReset `ara::diag::ResetRequestType` check** [The sub-function value shall be checked against the value of the `DiagnosticEcuReset.category` and passed in `resetType` parameter.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01019]{DRAFT} Custom `ara::diag::ResetRequestType` processing** [If the `resetType` is set to `kCustomReset`, the `id` shall be equal to the customer reset value of the request.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01020]{DRAFT} EnableRapidPowerShutdown processing** [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].] (*RS\_Diag\_04196*)

**[SWS\_DM\_01092]{DRAFT} EnableRapidPowerShutdown Positive Response** [If a positive response for subfunction 'EnableRapidPowerShutdown' shall be sent, the DM shall put the configured value of `SoftwareClusterDiagnosticDeploymentProps.powerDownTime` in its positive response.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01021]{DRAFT} DisableRapidPowerShutdown processing** [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].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00268]{DRAFT} EcuReset positive response processing before reset** [If the external processor did NOT raise an `ApApplicationError`, the `DiagnosticServer` instance shall 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`.] (*RS\_Diag\_04019*)

**[SWS\_DM\_01023]{DRAFT} Calling ExecuteReset() if positive response shall be sent before reset** [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`.] (*RS\_Diag\_04196*)

**[SWS\_DM\_00360]{DRAFT} EcuReset positive response processing after reset** [If the external processor did NOT raise an `ApApplicationError`, and `ara::diag::uds_transport::UdsTransportProtocolMgr::ChannelReestablished` was called and the parameter `DiagnosticEcuResetClass.respondToReset` is set to `DiagnosticResponseToEcuResetEnum.respondAfterReset`, the diagnostic Server instance shall send a positive response.] (*RS\_Diag\_04196*)

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

**[SWS\_DM\_01346]{DRAFT} Handling negative return values of `ara::diag::EcuResetRequest::ExecuteReset`** [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.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01347]{DRAFT} Handling unspecified negative return values of `ara::diag::EcuResetRequest::ExecuteReset`** [If `ara::diag::EcuResetRequest::ExecuteReset` returns any error code other than the defined error codes in [SWS\_DM\_01014], the Diagnostic Server instance shall return a Negative Response with NRC equal to 0x10 (`generalReject`), if a final response to the request was not sent yet.] (*RS\_Diag\_04196*)

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]{DRAFT} Calling ExecuteReset() if positive response shall be sent after reset** [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].] (*RS\_Diag\_04196*)

Note: Sending the positive response after the reset is handled by [SWS\_DM\_00360].

**[SWS\_DM\_00269]{DRAFT} Reaction on Unsupported Subfunction** [The Diagnostic Server instance shall send a negative response 0x12 (`SubFunctionNotSupported`), if the requested subfunction value is neither in configured range of default subfunction values (`requestType`, see ISO 14229-1[1]) nor in range of

the configured `DiagnosticEcuReset.customSubFunctionNumber` in the ECU.] ([RS\\_Diag\\_04196](#))

**[SWS\_DM\_01022]{DRAFT} Block requests after `ara::diag::EcuResetRequest::RequestReset` called** [If the reset request is accepted, the Diagnostic Server instance that received the EcuReset request shall ignore further incoming UDS requests.] ([RS\\_Diag\\_04196](#))

**[SWS\_DM\_01309]{DRAFT} Unblock requests after `ara::diag::EcuResetRequest::ExecuteReset` completed** [If UDS requests are blocked due to [\[SWS\\_DM\\_01022\]](#) and the `ara::core::Future` returned from the call of `ara::diag::EcuResetRequest::ExecuteReset`, the DM shall accept incoming UDS requests again.] ([RS\\_Diag\\_04196](#))

**[SWS\_DM\_01766]{DRAFT} Reporting 'ECU reset triggered ECUReset' to IdsM** [In case a diagnostic service `ECUReset` is received, which results in a positive response, the DM shall report a security event `SEV_UDS_ECU_RESET` to IdsM (see [\[11\]](#) and table [\[SWS\\_DM\\_01760\]](#)) with the context data given in [\[SWS\\_DM\\_01937\]](#)] ([RS\\_Ids\\_00810](#))

**[SWS\_DM\_01937]{DRAFT}** [

Context Data:	Subfunction of the requested ECU reset service
---------------	--

**Table 7.10: SEV on 'ECU reset triggered by EcuReset'**

]()

#### 7.4.2.8.6 Service 0x14 – ClearDiagnosticInformation

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

**[SWS\_DM\_00090]{DRAFT} Support of UDS service ClearDiagnosticInformation** [The `Diagnostic Server instance` shall provide the UDS service 0x14 Clear-DiagnosticInformation according to ISO 14229-1[1].] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04196](#))

**[SWS\_DM\_00091]{DRAFT} Evaluation of ClearDiagnosticInformation parameters** [The `Diagnostic Server instance` shall determine the `DTC group` or single `DTC` to clear from the 'groupOfDTC' parameter of the UDS request.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04117](#))

**[SWS\_DM\_00092]{DRAFT} Parameter range check for groupOfDTC request parameter** [The `Diagnostic Server instance` shall reply with an `NRC 0x31` (`RequestOutOfRange`) if the requested 'groupOfDTC' has no matching configured DTC group according to [\[SWS\\_DM\\_00064\]](#) or configured DTC by `DiagnosticTroubleCodeUds.udsDtcValue`.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04117](#))

**[SWS\_DM\_00113]{DRAFT} Positive response for UDS service 0x14** [If the *Diagnostic Server instance* has cleared the requested 'groupOfDTC', the *Diagnostic Server instance* shall send a positive response.]([RS\\_Diag\\_04196](#))

The DTC clearing behavior is described in detail in section 7.4.4.5.5. It consists of resetting the DTC status and deleting snapshot records and extended data records.

**[SWS\_DM\_01768]{DRAFT} Reporting 'DTC fault memory cleared' to IdsM** [In case a diagnostic service `ClearDTCInformation` is received, which results in a positive response, the *DM* shall report a security event `SEV_UDS_CLEAR_DTC_SUCCESSFUL` to *IdsM* (see [11] and table [\[SWS\\_DM\\_01760\]](#)) with the context data given in [\[SWS\\_DM\\_01938\]](#).]([RS\\_Ids\\_00810](#))

**[SWS\_DM\_01938]{DRAFT} [**

Context Data:	Memory type - memoryID (if available)
---------------	---------------------------------------

**Table 7.11: SEV on 'DTC fault memory cleared'**

**]()**

**[SWS\_DM\_00115]{DRAFT} Memory error handling while clearing DTCs** [The *Diagnostic Server instance* shall return a negative response `NRC 0x72` (`generalProgrammingFailure`) if it encounters an error in the non-volatile memory while clearing the DTCs.]([RS\\_Diag\\_04180](#))

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]{DRAFT} UDS response behavior on not allowed clear operations** [If a DTC clear operation is requested and the DTC clear operation shall clear a DTC with a forbidden clear allowance according to [\[SWS\\_DM\\_00896\]](#), the *Diagnostic Server instance* shall send a negative response `0x22` (`conditionsNotCorrect`) in the following situations:

- it was requested to clear a single DTC and the DTC could not be cleared according to [\[SWS\\_DM\\_00896\]](#)
- it was requested to clear a DTC group and all the DTCs of the DTC group could not be cleared according to [\[SWS\\_DM\\_00896\]](#)  
(This doesn't apply when one or more DTC are allowed to be cleared.)

**]([RS\\_Diag\\_04117](#))**

**[SWS\_DM\_00159]{DRAFT} Allow only to clear GroupOfAllDTCs** [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` (`RequestOutOfRange`).] ([RS\\_Diag\\_04117](#))

**[SWS\_DM\_00160]{DRAFT} Allow to clear single DTCs** [If the configuration `DiagnosticMemoryDestination.clearDtcLimitation` is set to `allSupportedDtc`s, the `Diagnostic Server instance` shall allow to clear single DTCs or DTC-Groups. [\[SWS\\_DM\\_00092\]](#) defines the possible and refused values.] ([RS\\_Diag\\_04117](#))

**[SWS\_DM\_01578]{DRAFT} Behavior of not configured DiagnosticMemoryDestination.clearDtcLimitation** [If the optional parameter `DiagnosticMemoryDestination.clearDtcLimitation` is not configured, the `DM` shall use a default value of `allSupportedDtc`s for that parameter.] ()

**[SWS\_DM\_00162]{DRAFT} Point in time for positive response for ClearDTC** [The `Diagnostic Server instance` shall send a positive response for a `ClearDiagnosticInformation` service after all memory is cleared in the server. This is regardless how the `Diagnostic Server instance` memory is organized (splitted, volatile, non-volatile).] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04196](#))

**[SWS\_DM\_00163]{DRAFT} Definition of a inhibited clear operation on single DTC** [If it is requested to clear a single DTC and a `DiagnosticClearCondition` referenced from this DTC (via `DiagnosticTroubleCodeUdsToClearConditionGroupMapping`) is not fulfilled, the `Diagnostic Server instance` shall send a negative response `0x22` (`conditionsNotCorrect`).] ([RS\\_Diag\\_04180](#))

**[SWS\_DM\_00164]{DRAFT} Definition of a inhibited clear operation for a group of DTCs** [If it is requested to clear a group of DTCs, the `Diagnostic Server instance` shall send a negative response `0x22` (`conditionsNotCorrect`) if all DTCs of that group of DTC forbid the clearance according to [\[SWS\\_DM\\_00163\]](#) or [\[SWS\\_DM\\_00896\]](#).] ([RS\\_Diag\\_04180](#))

#### 7.4.2.8.6.1 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]{DRAFT} Support of a user-defined fault memory clear request** [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`.] ([RS\\_Diag\\_04197](#))

**[SWS\_DM\_00194]{DRAFT} Definition of the user-defined fault memory number for ClearDiagnosticInformation** [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.](RS\_Diag\_04197)

**[SWS\_DM\_00195]{DRAFT} Clearing a user-defined memory** [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.](RS\_Diag\_04197)

For details about the fault memory clearing process please also refer to section 7.4.4.5.5.

**[SWS\_DM\_00208]{DRAFT} Validation of the requested user-defined memory number** [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)*.](RS\_Diag\_04197)

#### 7.4.2.8.7 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]{DRAFT} Mapping between ISO 14229-1 and Autosar Diagnostic Extract Template of the DTCFormatIdentifier** [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:](RS\_Diag\_04180, RS\_Diag\_04157, RS\_Diag\_04067)

<i>typeOfDtcSupported</i>	“DTCFormatIdentifier”
iso11992_4	0x03
iso14229_1	0x01
saeJ2012_da	0x00

**[SWS\_DM\_00966]{DRAFT} Reporting of DTCStatusAvailabilityMask** [For all positive response for service 0x19 with *DTCStatusAvailabilityMask* in the response,



the DM shall use the configured value from `dtcStatusAvailabilityMask`.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.1 SF 0x01 – reportNumberOfDTCByStatusMask

**[SWS\_DM\_00244]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x01** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

**[SWS\_DM\_00061]{DRAFT} Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask** [While sending the positive response for `ReadDTCInformation.reportNumberOfDTCByStatusMask`, the `Diagnostic Server instance` shall set the response PDU “DTCFormatIdentifier” according to the mapping of [\[SWS\\_DM\\_00062\]](#).] ([RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.2 SF 0x02 – reportDTCByStatusMask

**[SWS\_DM\_00245]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x02** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.3 SF 0x03 – reportDTCSnapshotIdentification

**[SWS\_DM\_01256]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x03** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.4 SF 0x04 – reportDTCSnapshotRecordByDTCNumber

**[SWS\_DM\_00246]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x04** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.5 SF 0x06 – reportDTCExtDataRecordByDTCNumber

**[SWS\_DM\_00370]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x06** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.6 SF 0x07 – reportNumberOfDTCBySeverityMaskRecord

**[SWS\_DM\_00247]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x07** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#))

**[SWS\_DM\_00063]{DRAFT} Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord** [While sending the positive response for ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord, the [Diagnostic Server instance](#) shall set the response PDU “DTCFormatIdentifier” according to the mapping of [\[SWS\\_DM\\_00062\]](#).] ([RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.7 SF 0x0A – reportSupportedDTC

[SWS\_DM\_00967]{DRAFT} **Support of UDS service ReadDTCInformation, Subfunction 0x0A** [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].] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

[SWS\_DM\_00968]{DRAFT} **Reporting of DTCAndStatusRecord parameter** [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.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.8 SF 0x14 – reportDTCFaultDetectionCounter

[SWS\_DM\_00371]{DRAFT} **Support of UDS service ReadDTCInformation, Subfunction 0x14** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.9 SF 0x17 – reportUserDefMemoryDTCByStatusMask

[SWS\_DM\_00372]{DRAFT} **Support of UDS service ReadDTCInformation, Subfunction 0x17** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.10 SF 0x18 – reportUserDefMemoryDTCSnapshotRecordByDTCNumber

[SWS\_DM\_00373]{DRAFT} **Support of UDS service ReadDTCInformation, Subfunction 0x18** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.7.11 SF 0x19 – reportUserDefMemoryDTCExtDataRecordByDTCNumber

**[SWS\_DM\_00374]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x19** [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'.] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04157](#), [RS\\_Diag\\_04067](#))

#### 7.4.2.8.8 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]{DRAFT} Realisation of UDS service ReadDataByIdentifier (0x22)** [The [Diagnostic Server instance](#) shall implement the diagnostic service 0x22 ReadDataByIdentifier according to ISO 14229-1[1].] ([RS\\_Diag\\_04196](#))

**[SWS\_DM\_00412]{DRAFT} Check requested number of DataIdentifiers** [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).] ([RS\\_Diag\\_04203](#))

**[SWS\_DM\_00409]{DRAFT} Check supported DataIdentifier** [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](#).] ([RS\\_Diag\\_04203](#))

**[SWS\_DM\_00413]{DRAFT} Check supported DataIdentifier in active session** [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.] ([RS\\_Diag\\_04203](#))

**[SWS\_DM\_00414]{DRAFT} Check supported DataIdentifier on active security level** [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.](*RS\_Diag\_04203*)

**[SWS\_DM\_00570]{DRAFT} Retrieving data for requested DataIdentifier** [On reception of the UDS Service ReadDataByIdentifier (0x22), the `Diagnostic Server instance` shall retrieve the mapped data for that DataIdentifier.

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

- received entire DID via `ara::diag` interface [SWS\_DM\_00601] or [SWS\_DM\_00607]
- received data for data elements via `ara::diag` interface [SWS\_DM\_00603]
- generic UDS service processing [SWS\_DM\_00602]
- internal data [SWS\_DM\_00393]

](*RS\_Diag\_04097*)

Note: Also, a `DiagnosticDataIdentifier`'s single `dataElement` (referenced by a meta-class `DiagnosticDataElement`) can be accessed from the associated `RPortPrototype`. Refer to chapters 7.4.6.2.1 and 7.4.6.2.2 for more details.

**[SWS\_DM\_00571]{DRAFT} Reaction on ApplicationError** [The Diagnostic Server instance shall derive negative response code based on the definition of ISO 14229-1[1] from `ara::diag::DiagUdsNrcErrorDomain`.](*RS\_Diag\_04196*)

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

#### 7.4.2.8.9 Service 0x27 – SecurityAccess

**[SWS\_DM\_00236]{DRAFT} Realization of UDS service 0x27 SecurityAccess** [The `Diagnostic Server instance` shall implement the diagnostic service 0x27 SecurityAccess according to ISO 14229-1[1.].](*RS\_Diag\_04196, RS\_Diag\_04005*)

**[SWS\_DM\_00863]{DRAFT} Checking Supported Subfunction for RequestSeed** [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. [\(RS\\_Diag\\_04203\)](#)

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]{DRAFT} Length check on UDS Service 0x27 request with Subfunction for RequestSeed** [On reception of a request for UDS Service `SecurityAccess` (0x27) with subfunction value matching the `requestSeedId` of a configured `DiagnosticSecurityAccess`, the `Diagnostic Server` instance shall perform the message length check against the optionally configured `accessDataRecordSize` of the related `DiagnosticSecurityLevel`. A non-present parameter `accessDataRecordSize` results in a check against 0 additional request bytes. If the length check fails, the `Diagnostic Server` instance shall send NRC 0x13 (IncorrectMessageLengthOrInvalidFormat).] [\(RS\\_Diag\\_04203\)](#)

**[SWS\_DM\_00864]{DRAFT} Checking Supported Subfunction for CompareKey** [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`.] [\(RS\\_Diag\\_04203\)](#)

**[SWS\_DM\_00363]{DRAFT} Unsupported Subfunction** [If the requested subfunction value is not configured (no instances of `DiagnosticSecurityAccess` with `requestSeedId`, as well as the corresponding `CompareKey` values), a negative response 0x12 (SubFunctionNotSupported) shall be returned.] [\(RS\\_Diag\\_04196\)](#)

**[SWS\_DM\_00846]{DRAFT} Notification about security-level change** [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`.] [\(RS\\_Diag\\_04208\)](#)

**[SWS\_DM\_01776]{DRAFT} Reporting 'security unlocked successfully for security access' to IdsM** [In case a diagnostic service `SecurityAccess` with subfunction `CompareKey` is received, which successfully unlocks the requested security access type, the `DM` shall report a security event `SEV_UDS_ECU_UNLOCK_SUCCESSFUL` to `IdsM` (see [11] and table [\[SWS\\_DM\\_01760\]](#)) with the context data given in [\[SWS\\_DM\\_01939\]](#).] [\(RS\\_Ids\\_00810\)](#)

**[SWS\_DM\_01939]{DRAFT}** [

Context Data:	subfunction (can be used to calculate the security level) of the requested security access service
---------------	--

**Table 7.12: SEV on 'security level unlocked successfully'**

]()

**[SWS\_DM\_00270]{DRAFT} Counting of attempts to change security level** [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.] (*RS\_Diag\_04005*)

**[SWS\_DM\_00271]{DRAFT} Evaluate the number of failed security level change attempts** [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.] (*RS\_Diag\_04005*)

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\_01775]{DRAFT} Reporting 'security access with invalid key' to IdsM** [In case a diagnostic service *SecurityAccess* with subfunction *CompareKey* is received, which is failing and the maximum number of allowed failed attempts not yet reached, the *DM* shall report a security event *SEV\_UDS\_SECURITY\_ACCESS\_INVALID\_KEY* to *IdsM* (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01940].] (*RS\_Ids\_00810*)

**[SWS\_DM\_01940]{DRAFT}** [

Context Data:	subfunction (can be used to calculate the security level) of the requested security access service
---------------	--

**Table 7.13: SEV on 'security access with invalid key'**

]()

**[SWS\_DM\_01774]{DRAFT} Reporting 'number of attempts exceeded for security access' to IdsM** [In case a diagnostic service *SecurityAccess* with subfunction *CompareKey* is received, which is failing and reaches the maximum number of allowed failed attempts, the *DM* shall report a security event *SEV\_UDS\_SECURITY\_ACCESS\_NUMBER\_OF\_ATTEMPTS\_EXCEEDED* to *IdsM* (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01941].] (*RS\_Ids\_00810*)

[SWS\_DM\_01941]{DRAFT} [

Context Data:	subfunction (can be used to calculate the security level) of the requested security access service
---------------	--

**Table 7.14: SEV on 'number of attempts exceeded for security access'**

]()

[SWS\_DM\_00272]{DRAFT} **Expiration of the delay timer** [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).

](RS\_Diag\_04005)

[SWS\_DM\_01773]{DRAFT} **Reporting 'required time delay not expired for security access' to IdsM** [In case a diagnostic service `SecurityAccess` with subfunction `RequestSeed` is received and the delay time is active, the DM shall report a security event `SEV_UDS_SECURITY_ACCESS_REQUIRED_TIME_DELAY_NOT_EXPIRED` to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01942].](RS\_Ids\_00810)

[SWS\_DM\_01942]{DRAFT} [

Context Data:	subfunction (can be used to calculate the security level) of the requested security access service
---------------	--

**Table 7.15: SEV on 'required time delay not expired for security access'**

]()

[SWS\_DM\_00478]{DRAFT} **Persistent Storage of failed attempts to change security level** [The `Diagnostic Server instance` shall store the number of failed attempts persistently for every security access type separately. (see [SWS\_DM\_00270])](RS\_Diag\_04005)

[SWS\_DM\_00479]{DRAFT} **Security Access Delay Timer on power up when attempt counter threshold is reached** [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`.](RS\_Diag\_04005)

[SWS\_DM\_01758]{DRAFT} **Security Access Delay Timer on power up** [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`.](RS\_Diag\_04005)



**[SWS\_DM\_00480]{DRAFT} Shared Security Access Delay Timer** [If `DiagnosticSecurityAccessClass.sharedTimer` is set to true, a shared delay timer instance shall be used for all security levels.] (*RS\_Diag\_04005*)

The effect of a started security access delay timer is always according to [SWS\_DM\_00272].

#### 7.4.2.8.10 Service 0x28 – CommunicationControl

**[SWS\_DM\_00140]{DRAFT} Realisation of UDS service 0x28 CommunicationControl** [The `Diagnostic Server instance` shall implement the diagnostic service 0x28 CommunicationControl according to ISO 14229-1[1].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00252]{DRAFT} Reaction on Unsupported Subfunction** [The `Diagnostic Server instance` shall check, whether the Subfunction addressed by the CommunicationControl is supported by an existing `DiagnosticComControl.category` in the configuration and allow further processing. If the Subfunction addressed by the CommunicationControl is not supported by an existing `DiagnosticComControl.category` in the configuration a negative response 0x12 (SubFunctionNotSupported) shall be returned.] (*RS\_Diag\_04203*)

**[SWS\_DM\_00865]{DRAFT} Communication control service processing** [The `Diagnostic Server instance` shall call the method `ara::diag::CommunicationControl::CommCtrlRequest` to process a communication control service.] (*RS\_Diag\_04169*)

**[SWS\_DM\_00866]{DRAFT} Negative Response processing** [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.] (*RS\_Diag\_04196*)

**[SWS\_DM\_00199]{DRAFT} Positive Response processing** [If the external processor did raise no `ApApplicationError`, the `Diagnostic Server instance` shall return a positive response.] (*RS\_Diag\_04196*)

**[SWS\_DM\_01767]{DRAFT} Reporting 'Communication control switched off' to IdsM** [In case a diagnostic service CommunicationControl with subfunction either `enableRxAndDisableTx`, `disableRxAndTx` or `enableRxAndDisableTxWithEnhancedAddressInformation` is received, which results in a positive response, the DM shall report a security event SEV\_UDS\_COMMUNICATION\_CONTROL\_SWITCHED\_OFF to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01943].] (*RS\_Ids\_00810*)

**[SWS\_DM\_01943]{DRAFT}** [

Context Data:	Subfunction of the requested communication control service
---------------	--

**Table 7.16: SEV on 'Communication control switched off'**

]0

#### 7.4.2.8.11 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]{DRAFT} Support of UDS service authentication** [If configured, the [Diagnostic Server instance](#) shall provide the UDS service 0x29 Authentication with the subfunctions.

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

according to ISO 14229-1:2020 [1].] ([RS\\_Diag\\_04251](#))

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]{DRAFT} Configuration of authentication types** [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](#)

] ([RS\\_Diag\\_04251](#))

**[SWS\_DM\_01228]{DRAFT} Mandatory sub functions** [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

](RS\_Diag\_04251)

**[SWS\_DM\_01771]{DRAFT} Reporting 'certificate failed' to IdsM** [In case a diagnostic service Authentication with subfunction verifyCertificateUnidirectional, verifyCertificateBidirectional or proofOfOwnership is received, which fails in the certification verification along [SWS\_DM\_01231], [SWS\_DM\_01236] or [SWS\_DM\_01241], the DM shall report a security event SEV\_UDS\_CERTIFICATE\_FAILURE to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01944].](RS\_Ids\_00810)

**[SWS\_DM\_01944]{DRAFT}** [

Context Data:	Client source address (Metainfo kSA), who requested the authentication
---------------	--

**Table 7.17: SEV on 'certificate failed'**

]()

#### 7.4.2.8.11.1 VerifyCertificateUnidirectional

**[SWS\_DM\_01230]{DRAFT} Processing the verifyCertificateUnidirectional request** [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.](RS\_Diag\_04251)

**[SWS\_DM\_01231]{DRAFT} Handling Negative return values of `ara::diag::Authentication::VerifyCertificateUnidirectional` method** [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.](RS\_Diag\_04251)

**[SWS\_DM\_01233]{DRAFT} Successful verification of verifyCertificateUnidirectional** [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. *(RS\_Diag\_04251)*

**[SWS\_DM\_01234]{DRAFT} Unexpected verifyCertificateUnidirectional from a different client** [The method `ara::diag::Authentication::VerifyCertificateUnidirectional` shall be re-entrant to requests from different clients.] *(RS\_Diag\_04251)*

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

#### 7.4.2.8.11.2 VerifyCertificateBidirectional

**[SWS\_DM\_01235]{DRAFT} Processing the verifyCertificateBidirectional request** [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.] *(RS\_Diag\_04251)*

**[SWS\_DM\_01236]{DRAFT} Handling Negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method** [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.] *(RS\_Diag\_04251)*

**[SWS\_DM\_01238]{DRAFT} Successful verification of `verifyCertificateBidirectional`** [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 (POWNSE) and the returned Ephemeral Public Key into the response field ephemeralPublicKeyServer (EPKSE). The *Diagnostic Server instance* shall set the parameter authenticationReturnParameter (RV) to 0x11 (CertificateVerified, OwnershipVerificationNecessary) in the positive Response.](RS\_Diag\_04251)

**[SWS\_DM\_01239]{DRAFT} Unexpected verifyCertificateBidirectional from a different client** [The method `ara::diag::Authentication::VerifyCertificateBidirectional` shall be re-entrant to requests from different clients.](RS\_Diag\_04251)

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

#### 7.4.2.8.11.3 ProofOfOwnership

**[SWS\_DM\_01240]{DRAFT} Processing the proofOfOwnership request** [On reception of an Authentication (0x29) service with sub-function equal to `proofOfOwnership`, if all checks described in [SWS\_DM\_00096] are successfully completed, the *Diagnostic Server instance* shall call the method `ara::diag::Authentication::VerifyOwnership` of the class `ara::diag::Authentication`. The *Diagnostic Server instance* shall pass the received `proofOfOwnershipClient` (POWNCL) and `ephemeralPublicKeyClient` (EPKCL) to the application in the parameters `ClientPOWN` and `ClientEphemeralPublicKey` of the method respectively.](RS\_Diag\_04251)

**[SWS\_DM\_01241]{DRAFT} Handling Negative return values of `ara::diag::Authentication::VerifyOwnership` method** [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.](RS\_Diag\_04251)

**[SWS\_DM\_01243]{DRAFT} Successful verification of Client `proofOfOwnership`** [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.](RS\_Diag\_04251)

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.

**[SWS\_DM\_01770]{DRAFT} Reporting 'successful authentication' to IdsM** [In case a diagnostic service *Authentication* with subfunction *proofOfOwnership* is received, which is successful along [SWS\_DM\_01243], the *DM* shall report a security event SEV\_UDS\_AUTHENTICATION\_SUCCESSFUL to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01945].] (*RS\_IdS\_00810*)

**[SWS\_DM\_01945]{DRAFT}** [

Context Data:	Client source address (Metainfo kSA), who requested the authentication
---------------	--

**Table 7.18: SEV on 'successful authentication'**

]()

#### 7.4.2.8.11.4 DeAuthenticate

**[SWS\_DM\_01244]{DRAFT} Processing the deAuthenticate request** [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].] (*RS\_Diag\_04251*)

**[SWS\_DM\_01245]{DRAFT} Successful completion of deAuthenticate** [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.] (*RS\_Diag\_04251*)

#### 7.4.2.8.11.5 AuthenticationConfiguration

**[SWS\_DM\_01246]{DRAFT} Processing the authenticationConfiguration request** [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.] (*RS\_Diag\_04251*)

#### 7.4.2.8.11.6 TransmitCertificate

**[SWS\_DM\_01247]{DRAFT} Validation of the transmitCertificate certificateEvaluationId** [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).] (*RS\_Diag\_04251*)

**[SWS\_DM\_01248]{DRAFT} Processing the transmitCertificate request** [If the checks described in [SWS\_DM\_01247] are successfully completed, the *Diagnostic Server instance* shall call the method `ara::diag::Authentication::TransmitCertificate` of the class `ara::diag::Authentication`. The *Diagnostic Server instance* shall pass the received certificateData (CEDA) in the parameter CertificateData of the method.] (*RS\_Diag\_04251*)

**[SWS\_DM\_01249]{DRAFT} Handling Negative return values of ara::diag::Authentication::TransmitCertificate method** [If the method called in [SWS\_DM\_01248] returns any of the defined error codes in [SWS\_DM\_01129], the *Diagnostic Server instance* shall return a Negative Response with NRC equal to the returned error code.] (*RS\_Diag\_04251*)

**[SWS\_DM\_01251]{DRAFT} Successful verification of transmitCertificate** [If the method called in [SWS\_DM\_01248] executes successfully and returns no error code, the *Diagnostic Server instance* shall return a Positive Response to the tester. The *Diagnostic Server instance* shall set the parameter authenticationReturnParameter (RV) to 0x13 (CertificateVerified) in the positive Response.] (*RS\_Diag\_04251*)

#### 7.4.2.8.12 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]{DRAFT} Realization of UDS service ReadDataByPeriodicIdentifier(0x2A)** [The *Diagnostic Server instance* shall implement the diagnostic service 0x2A ReadDataByPeriodicIdentifier according to ISO 14229-1[1].] (*RS\_Diag\_04215*)

**[SWS\_DM\_01041]{DRAFT} Check requested number of periodic DataIdentifiers** [If the DM receives the UDS Service ReadDataByPeriodicIdentifier (0x2A) and the

number of the requested PeriodicDataIdentifiers is larger than `DiagnosticReadDataByPeriodicIDClass.maxPeriodicDidToRead`, the DM shall return NRC 0x13 (incorrectMessageLengthOrInvalidFormat).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01042]{DRAFT} Minimum length check for ReadDataByPeriodicIdentifier** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), the DM shall check the request minimum length. If length of the request is less than 3 bytes for subfunctions different to 'stopSending' or less than 2 bytes, the DM shall respond with NRC 0x13 (incorrectMessageLengthOrInvalidFormat).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01043]{DRAFT} Check supported periodic DataIdentifier** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A), the DM shall consider a requested PeriodicDataIdentifier as supported if and only if there exists a DiagnosticDataIdentifier in the range between 0xF200 and 0xF2FF with id matching the PeriodicDataIdentifier. If none of the requested Periodic DIDs are supported, the DM shall respond with NRC 0x31 (requestOutOfRange).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01044]{DRAFT} Check Transmission Mode** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) and if the requested transmission mode is different to a configured `DiagnosticPeriodicRate.periodicRateCategory` or to 'stopSending', the DM shall respond with NRC 0x31 (requestOutOfRange).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01045]{DRAFT} Check Scheduler Availability** [If the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode different than 0x04 'stopSending' is received, and the number existing PDIDs and the new PDIDs from the request is larger than `DiagnosticReadDataByPeriodicIDClass.schedulerMaxNumber`, the DM shall respond with NRC 0x31 (requestOutOfRange).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01046]{DRAFT} Check supported DataIdentifier in active session** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than 'stopSending', a requested Periodic DataIdentifier shall be considered as supported if and only if the active session passes the execution permission check according to [\[SWS\\_DM\\_00413\]](#) else process next periodic DID. If Session not supported for none of the periodic DIDs DM shall respond with NRC 0x31 (requestOutOfRange).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01047]{DRAFT} Check supported DataIdentifier on active security level** [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).] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01048]{DRAFT} Check DataIdentifier for environmental conditions** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmission-Mode different than stopSending, a requested PeriodicDataIdentifier shall be considered as supported if and only if the DataIdentifiers environmentalCondition allow an



execution according to [SWS\_DM\_00112] else the DM shall respond with the NRC according to [SWS\_DM\_00289].] (*RS\_Diag\_04215*)

**[SWS\_DM\_01049]{DRAFT} Checks Dynamically Defined DIDs in ReadDataByPeriodicIdentifier** [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.] (*RS\_Diag\_04215*)

**[SWS\_DM\_01050]{DRAFT} Periodic DID length check** [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 `ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength` and if the size is exceeded for one or more PeriodicDataIdentifier, the DM shall return NRC 0x14 (responseTooLong).] (*RS\_Diag\_04215*)

**[SWS\_DM\_01051]{DRAFT} DM behavior on transmission Mode stopSending without periodicDataIdentifier in the request** [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.] (*RS\_Diag\_04215*)

**[SWS\_DM\_01052]{DRAFT} DM behavior on transmission Mode stopSending with supported periodicDataIdentifier in the request** [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.] (*RS\_Diag\_04215*)

**[SWS\_DM\_01053]{DRAFT} DM behavior on transmission Mode stopSending with not supported periodicDataIdentifier in the request** [On reception of the UDS Service ReadDataByPeriodicIdentifier (0x2A) with transmissionMode set to 'stopSending' and none of the periodicDataIdentifiers of the request is supported, the DM shall return NRC 0x31 (requestOutOfRange).] (*RS\_Diag\_04215*)

**[SWS\_DM\_01054]{DRAFT} Starting to transmit PDIDs after positive response** [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.] (*RS\_Diag\_04215*)

**[SWS\_DM\_01055]{DRAFT} Reaction on ApplicationError** [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.] (*RS\_Diag\_04215*)

**[SWS\_DM\_01056]{DRAFT} Optional condition checks for sending periodic DIDs** [If `DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId` is set to TRUE, the DM shall transmit the PDID only if the session, security and mode checks were executed successfully.] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01057]{DRAFT} Optional stopping PDIDs after session change** [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.] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01058]{DRAFT} Optional stopping PDIDs after security level change** [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.] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01059]{DRAFT} No periodic DIDs in default session** [When the diagnostic session changes to or is in the DefaultSession, the DM shall stop all scheduled periodic DIDs.] ([RS\\_Diag\\_04215](#))

### 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]{DRAFT} Support of Scheduler type 1** [The DM shall support the scheduling of PDIDs based on scheduler type 1 as defined in ISO 14229-1[1].] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01061]{DRAFT} Trigger all scheduled PDIDs per scheduler** [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.] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01062]{DRAFT} Transmission of all PDIDs on the periodic connection** [If the PDID transmission is triggered for a requested transmission rate, the DM shall transmit all PDIDs on the periodic transmission on the periodic connection starting with the first PDID in the list of scheduled PDIDs.] ([RS\\_Diag\\_04215](#))

**[SWS\_DM\_01063]{DRAFT} Transmission error behavior** [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.] ([RS\\_Diag\\_04215](#))

#### 7.4.2.8.13 Service 0x2C – DynamicallyDefineDataIdentifier

**[SWS\_DM\_01070]{DRAFT} Support of UDS service 0x2C in Adaptive AUTOSAR DM** [The *Diagnostic Server instance* shall implement the diagnostic service 0x2C DynamicallyDefineDataIdentifier with subfunctions 0x01 (defineByIdentifier) and 0x03 (clearDynamicallyDefinedDataIdentifier) according to ISO 14229-1[1].] (*RS\_Diag\_04246*)

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

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

**[SWS\_DM\_01071]{DRAFT} No persistency of defined DDDIDs** [If *DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling* is set to volatile, the *DM* shall initialize all *DDDIDs* as not present at *DM* Initialization.] (*RS\_Diag\_04246*)

**[SWS\_DM\_01072]{DRAFT} Persistency of defined DDDIDs** [If *DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling* is set to non-Volatile, the *DM* shall persist *DDDIDs* configured by the UDS service 0x2C and restore the *DDDIDs* definition from Non-Volatile Memory at *DM* Initialization.] (*RS\_Diag\_04246*)

**[SWS\_DM\_01073]{DRAFT} DM behavior for subfunction 'defineByIdentifier'** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier' the *DM* shall activate this *DDDID* and append the associated information received from the diagnostic request: *DID* source, position and size.] (*RS\_Diag\_04246*)

**[SWS\_DM\_01074]{DRAFT} Only static IDs as sourceDataIdentifier** [If the *DM* is receiving the service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier' and one or more of the sourceDataIdentifier are itself a active dynamically defined data identifier, the *DM* shall return an *NRC* 0x31 (requestOutOfRange).] (*RS\_Diag\_04246*)

**[SWS\_DM\_01075]{DRAFT} Maximum number of sourceDataIdentifiers in the request** [If the *DM* receives the UDS service 'DynamicallyDefineDataIdentifier' (0x2C) and the number of sourceDataIdentifiers in the request is larger than the configured *DiagnosticDynamicallyDefineDataIdentifier.maxSourceElement*, the *DM* shall send *NRC* 0x31 (requestOutOfRange).] (*RS\_Diag\_04246*)

**[SWS\_DM\_01076]{DRAFT} Clearing all configured DDDIDs** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clearDynamicallyDefinedDataIdentifier and no dynamicallyDefinedDataIdentifier in the request, the *DM* shall clear all the configured *DDDIDs*.] (*RS\_Diag\_04246*)

**[SWS\_DM\_01077]{DRAFT} Clearing individual configured DDDIDs** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 clearDynamically-

DefinedDataIdentifier and one dynamicallyDefinedDataIdentifier in the request, the DM shall clear the according DDDID from the request.](RS\_Diag\_04246)

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

**[SWS\_DM\_01078]{DRAFT} Clear DDDIDs on session change** [If DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to true and a diagnostic session change occurs, the DM shall clear all defined DDDIDs that have sourceDataIdentifiers that are no longer supported in the new session and security level.](RS\_Diag\_04246)

**[SWS\_DM\_01079]{DRAFT} Session check for DDDID** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDDID can be defined in the current session according to [SWS\_DM\_00413]. If the DDDID cannot be defined in the current session, the DM shall return an NRC 0x31 (requestOutOfRange).](RS\_Diag\_04246)

**[SWS\_DM\_01080]{DRAFT} Security level check for DDDID** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDDIDs can be defined in the security level according to [SWS\_DM\_00414]. If the DDDID cannot be defined in the current security level, the DM shall return an NRC 0x31 (requestOutOfRange).](RS\_Diag\_04246)

**[SWS\_DM\_01081]{DRAFT} Session check for sourceDataIdentifier** [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction defineByIdentifier and if DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId is set to true, the DM shall check if the sourceDataIdentifier can be defined in the current session according to [SWS\_DM\_00413]. If any sourceDataIdentifier cannot be defined in the current session, the DM shall return an NRC 0x31 (requestOutOfRange).](RS\_Diag\_04246)

**[SWS\_DM\_01082]{DRAFT} Security level check for sourceDataIdentifier** [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).](RS\_Diag\_04246)

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

**[SWS\_DM\_01083]{DRAFT} Use of configured DID ports to get DDDID data** [If a DDDID read operation is requested, the DM shall retrieve the data for the dataRecord of the DDDID by using the configuration of the contained DIDs.](RS\_Diag\_04246)

#### 7.4.2.8.14 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]{DRAFT} Realisation of UDS service WriteDataByIdentifier (0x2E)** [The Diagnostic Server instance shall implement the diagnostic service 0x2E WriteDataByIdentifier according to ISO 14229-1[1].] (RS\_Diag\_04196)

**[SWS\_DM\_00415]{DRAFT} Check supported DataIdentifier** [On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported if and only if there exists a DiagnosticDataIdentifier with id matching the DataIdentifier and this DiagnosticDataIdentifier is referenced by a DiagnosticWriteDataByIdentifier.] (RS\_Diag\_04203)

**[SWS\_DM\_00416]{DRAFT} Check supported DataIdentifier in active session** [On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported in active session if and only if the DataIdentifier is supported according to [SWS\_DM\_00415] and the active session passes the execution permission check as per [SWS\_DM\_00101].] (RS\_Diag\_04203)

**[SWS\_DM\_00417]{DRAFT} Check supported DataIdentifier on active security level** [On reception of the UDS Service WriteDataByIdentifier (0x2E), a requested DataIdentifier shall be considered as supported on active security level if and only if the DataIdentifier is supported according to [SWS\_DM\_00415] and the active security level passes the execution permission check as per [SWS\_DM\_00103].] (RS\_Diag\_04203)

**[SWS\_DM\_00572]{DRAFT} Writing data for requested DataIdentifier** [On reception of the UDS Service WriteDataByIdentifier (0x2E) the Diagnostic Server instance shall provide the data for a DataIdentifier to the mapped RPortPrototypes.] (RS\_Diag\_04097)

**[SWS\_DM\_00573]{DRAFT} Reaction on ApplicationError** [If the *ara::core::Result* of external processor has an error of *ara::diag::DiagUdsNrcErrorDomain*, the Diagnostic Server instance shall return a negative response with the value of that error code.] (RS\_Diag\_04196)

**[SWS\_DM\_01764]{DRAFT} Reporting 'Writing invalid data is requested' to IdsM** [In case a diagnostic service WriteDataByIdentifier is received, which results in a negative response with NRC 0x31 (*requestOutOfRange*), the DM shall report a security event SEV\_UDS\_WRITE\_INVALID\_DATA to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01946].] (RS\_Ids\_00810)

[SWS\_DM\_01946]{DRAFT} [

Context Data:	Requested DID
---------------	---------------

Table 7.19: SEV on 'Writing invalid data is requested'

]()

[SWS\_DM\_01763]{DRAFT} **Reporting 'Data written using WriteDataByIdentifier' to IdsM** [In case a diagnostic service `WriteDataByIdentifier` is received, which results in a positive response, the `DM` shall report a security event `SEV_UDS_WRITE_DATA` to `IdsM` (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01947].] (*RS\_Ids\_00810*)

[SWS\_DM\_01947]{DRAFT} [

Context Data:	Requested DID
---------------	---------------

Table 7.20: SEV on 'Data written using WriteDataByIdentifier'

]()

#### 7.4.2.8.15 Service 0x31 – RoutineControl

[SWS\_DM\_00201]{DRAFT} **Realization of UDS service RoutineControl (0x31)** [The `Diagnostic Server instance` shall implement the diagnostic service `RoutineControl` (0x31) according to ISO 14229-1[1] for subFunctions `startRoutine`, `stopRoutine` and `requestRoutineResults`.] (*RS\_Diag\_04196*, *RS\_Diag\_04224*)

[SWS\_DM\_00202]{DRAFT} **Check for Supported RoutineIdentifier and Reaction** [The `Diagnostic Server instance` shall check, whether the `RoutineIdentifier` addressed by the `UDS Service RoutineControl` (0x31) is supported by an existing `DiagnosticRoutine` with a matching `id` in the configuration. If the `RoutineIdentifier` addressed by the `UDS Service RoutineControl` (0x31) is not supported a negative response with `NRC 0x31` (`requestOutOfRange`) shall be returned.] (*RS\_Diag\_04203*, *RS\_Diag\_04224*)

[SWS\_DM\_00448]{DRAFT} **Check supported RoutineIdentifier in active session** [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.] (*RS\_Diag\_04203*, *RS\_Diag\_04224*)

[SWS\_DM\_00437]{DRAFT} **Security Level check for RoutineIdentifier** [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.] (*RS\_Diag\_04203*, *RS\_Diag\_04224*)

**[SWS\_DM\_00203]{DRAFT} Check for Supported Subfunction and Reaction** [The *Diagnostic Server instance* shall check, whether the Subfunction addressed by the UDS Service RoutineControl (0x31) is supported by checking the existence of the corresponding attributes *start* or *stop* or *requestResult* in the related *DiagnosticRoutine* configuration. If the Subfunction addressed by the UDS Service RoutineControl (0x31) is not supported by the configuration a negative response *NRC 0x12* (*SubFunctionNotSupported*) shall be returned.](*RS\_Diag\_04203*, *RS\_Diag\_04224*)

**[SWS\_DM\_00574]{DRAFT} UDS Service RoutineControl (0x31) startRoutine processing with generic interface** [The *Diagnostic Server instance* shall call the *ara::diag::GenericRoutine::Start* according to the mapped diagnostic interface [TPS\_MANI\_01326] and [constr\_10062] to process the subfunction *startRoutine*.](*RS\_Diag\_04196*, *RS\_Diag\_04224*)

**[SWS\_DM\_00575]{DRAFT} UDS Service RoutineControl (0x31) requestRoutineResults processing with generic interface** [The *Diagnostic Server instance* shall call *ara::diag::GenericRoutine::RequestResults* according to the mapped diagnostic interface [TPS\_MANI\_01326] and [constr\_10062] to process the subfunction *requestRoutineResults*.](*RS\_Diag\_04196*, *RS\_Diag\_04224*)

**[SWS\_DM\_00576]{DRAFT} UDS Service RoutineControl (0x31) stopRoutine processing with generic interface** [The *Diagnostic Server instance* shall call *ara::diag::GenericRoutine::Stop* according to the mapped diagnostic interface [TPS\_MANI\_01326] and [constr\_10062] to process the subfunction *stopRoutine*.](*RS\_Diag\_04196*, *RS\_Diag\_04224*)

**[SWS\_DM\_01270]{DRAFT} UDS Service RoutineControl (0x31) startRoutine processing with typed interface** [The *Diagnostic Server instance* shall utilize the associated *RPortPrototype* if it is typed by the *Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface* class and call its *Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Start* function to process the subfunction *startRoutine*.](*RS\_Diag\_04196*, *RS\_Diag\_04224*)

**[SWS\_DM\_01271]{DRAFT} UDS Service RoutineControl (0x31) stopRoutine processing with typed interface** [The *Diagnostic Server instance* shall utilize the associated *RPortPrototype* if it is typed by the *Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface* class and call its *Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Stop* function to process the subfunction *stopRoutine*.](*RS\_Diag\_04196*, *RS\_Diag\_04224*)

**[SWS\_DM\_01272]{DRAFT} UDS Service RoutineControl (0x31) requestRoutineResults processing with typed interface** [The *Diagnostic Server instance* shall utilize the associated *RPortPrototype* if it is typed by the *Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface* class and call its *Namespace1OfPortInterface::*

`Namespace2OfPortInterface::ShortnameOfRIPortInterface::RequestResults` function to process the subfunction `requestRoutineResults`.] (*RS\_Diag\_04196, RS\_Diag\_04224*)

**[SWS\_DM\_01273]{DRAFT} Namespace for typed UDS Service RoutineControl (0x31)** [If the typed RoutineControl interface is used, it shall fulfill **[SWS\_DM\_00510]** also.] (*RS\_Diag\_04196, RS\_Diag\_04224*)

#### 7.4.2.8.16 Service 0x34 – RequestDownload

**[SWS\_DM\_00128]{DRAFT} Realization of UDS service RequestDownload (0x34)** [The *Diagnostic Server instance* shall implement the UDS service RequestDownload (0x34) according to ISO 14229-1[1].] (*RS\_Diag\_04196, RS\_Diag\_04033*)

**[SWS\_DM\_00867]{DRAFT} UDS service RequestDownload (0x34) processing** [The *Diagnostic Server instance* shall call `ara::diag::DownloadService::RequestDownload` to process an UDS service RequestDownload (0x34).] (*RS\_Diag\_04196*)

**[SWS\_DM\_01765]{DRAFT} Reporting 'Download sequence requested' to IdsM** [In case a diagnostic service `RequestDownload` is received, which results in a positive response, the *DM* shall report a security event `SEV_UDS_REQUEST_DOWNLOAD` to *IdsM* (see [11] and table **[SWS\_DM\_01760]**) with the context data given in **[SWS\_DM\_01948]**.] (*RS\_Ids\_00810*)

**[SWS\_DM\_01948]{DRAFT}** [

Context Data:	no data provided
---------------	------------------

**Table 7.21: SEV on 'Download sequence requested'**

]()

#### 7.4.2.8.17 Service 0x35 – RequestUpload

**[SWS\_DM\_00134]{DRAFT} Realization of UDS service RequestUpload (0x35)** [The *Diagnostic Server instance* shall implement the UDS service RequestUpload (0x35) according to ISO 14229-1[1].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00868]{DRAFT} UDS service RequestUpload (0x35) processing** [The *Diagnostic Server instance* shall call `ara::diag::UploadService::RequestUpload` to process a UDS service RequestUpload (0x35).] (*RS\_Diag\_04033*)



#### 7.4.2.8.18 Service 0x36 – TransferData

**[SWS\_DM\_00137]{DRAFT} Realization of UDS service TransferData (0x36)** [The *Diagnostic Server instance* shall implement the UDS service TransferData (0x36) according to ISO 14229-1[1].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00869]{DRAFT} UDS service TransferData (0x36) download processing** [To download data via the UDS service TransferData, the *Diagnostic Server instance* shall call `ara::diag::DownloadService::DownloadData` to process the service.] (*RS\_Diag\_04033*)

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 “*GenericUDSService class*” has only one single method this is not possible for the “*GenericUDSService class*”. As a result of this, the entire service specific NRC handling is inside the “*GenericUDSService class*” for UDS service TransferData (0x36).

**[SWS\_DM\_01096]{DRAFT} UDS service TransferData (0x36) upload processing** [To upload data via the UDS service TransferData, the *Diagnostic Server instance* shall call `ara::diag::UploadService::UploadData` to process the service.] (*RS\_Diag\_04033*)

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].

#### 7.4.2.8.19 Service 0x37 – RequestTransferExit

**[SWS\_DM\_00141]{DRAFT} Realization of UDS service RequestTransferExit (0x37)** [The *Diagnostic Server instance* shall implement the UDS service RequestTransferExit (0x37) according to ISO 14229-1[1].] (*RS\_Diag\_04196*)

**[SWS\_DM\_00871]{DRAFT} UDS service RequestTransferExit (0x37) download processing** [The *Diagnostic Server instance* shall call `ara::diag::DownloadService::RequestDownloadExit` to process a UDS service RequestTransferExit (0x37) for a download.] (*RS\_Diag\_04033*)

**[SWS\_DM\_01097]{DRAFT} UDS service RequestTransferExit (0x37) upload processing** [The *Diagnostic Server instance* shall call `ara::diag::UploadService::RequestUploadExit` to process a UDS service RequestTransferExit (0x37) for an upload.] (*RS\_Diag\_04033*)

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 [\[SWS\\_DM\\_01097\]](#).

#### 7.4.2.8.20 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]{DRAFT} Realization of UDS service RequestFileTransfer (0x38)** [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.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01311]{DRAFT} Realization of modeOfOperation AddFile (0x01)** [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.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01312]{DRAFT} Realization of modeOfOperation DeleteFile (0x02)** [If a service RequestFileTransfer with modeOfOperation equal to DeleteFile (0x02) is received, the [Diagnostic Server instances](#) shall call [ara::diag::FileTransferService::DeleteFile](#).] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01313]{DRAFT} Realization of modeOfOperation ReplaceFile (0x03)** [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.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01314]{DRAFT} Realization of modeOfOperation ReadFile (0x04)** [If a service RequestFileTransfer with modeOfOperation equal to ReadFile (0x04) is received, the *Diagnostic Server instance* shall call `ara::diag::FileTransferService::RequestReadFile`.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01315]{DRAFT} Realization of modeOfOperation ReadDir (0x05)** [If a service RequestFileTransfer with modeOfOperation equal to ReadDir (0x05) is received, the *Diagnostic Server instance* shall call `ara::diag::FileTransferService::RequestReadDirectory`.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01316]{DRAFT} Realization of modeOfOperation ResumeFile (0x06)** [If a service RequestFileTransfer with modeOfOperation equal to ResumeFile (0x06) is received, the *Diagnostic Server instance* shall call `ara::diag::FileTransferService::RequestResumeWriteFile`.] ([RS\\_Diag\\_04135](#))

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]{DRAFT} Realization of TransferData (0x36) in context of RequestFileTransfer** [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`.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01318]{DRAFT} Realization of RequestTransferExit (0x37) in context of RequestFileTransfer** [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`.] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01319]{DRAFT} Consecutive registration of notifier with ReleaseHandler::SetNotifier()** [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.] ([RS\\_Diag\\_04135](#))

## 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`.

### 7.4.2.8.21 Service 0x3E – TesterPresent

**[SWS\_DM\_00126]{DRAFT} Realisation of UDS service 0x3E TesterPresent** [The `Diagnostic Server instance` shall internally implement the diagnostic service `0x3E TesterPresent` according to ISO 14229-1[1].] ([RS\\_Diag\\_04196](#))

### 7.4.2.8.22 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.

**[SWS\_DM\_00229]{DRAFT} Support of UDS service ControlDTCSetting (0x85)** [The `Diagnostic Server instance` shall provide the `UDS` service `ControlDTCSetting (0x85)` according to ISO 14229-1[1].] ([RS\\_Diag\\_04180](#), [RS\\_Diag\\_04159](#))

**[SWS\_DM\_00444]{DRAFT} Check Support of UDS service ControlDTCSetting (0x85) in active session** [On reception of the `UDS` service `ControlDTCSetting (0x85)`, a requested subfunction shall be considered as supported in active session if and only

if the active session passes the execution permission check as per [SWS\_DM\_00101].]  
(RS\_Diag\_04203)

**[SWS\_DM\_00445]{DRAFT} Check Support of UDS service ControlDTCSetting (0x85) on active security level** [On reception of the UDS service ControlDTCSetting (0x85), a requested subfunction shall be considered as supported on active security level if and only if the active security level passes the execution permission check as per [SWS\_DM\_00103].](RS\_Diag\_04203)

**[SWS\_DM\_00230]{DRAFT} Check for supported subfunctions** [If the Subfunction addressed by the UDS service ControlDTCSetting (0x85) according to [SWS\_DM\_00229] is not Subfunction 0x01 (ON) or Subfunction 0x02 (OFF), then the Diagnostic Server instance shall return a NRC 0x12 (SubFunctionNotSupported).](RS\_Diag\_04203)

**[SWS\_DM\_00231]{DRAFT} Invalid value for optional request parameter** [If the Diagnostic Server instance receives a UDS service ControlDTCSetting (0x85) request with DTCSettingControlOptionRecord != 0xFFFFFFFF, the Diagnostic Server instance shall send a NRC 0x31 (RequestOutOfRange).](RS\_Diag\_04203, RS\_Diag\_04115)

**[SWS\_DM\_00909]{DRAFT} Support of Subfunction 0x01 (ON)** [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 primary event memory
- enable the storage in the primary event memory
- update `ara::diag::ControlDtcStatusType` to `kDTCSettingOn`

](RS\_Diag\_04180, RS\_Diag\_04159)

**[SWS\_DM\_00910]{DRAFT} Support of Subfunction 0x02 (OFF)** [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
- disable the storage in the primary event memory
- update `ara::diag::ControlDtcStatusType` to `kDTCSettingOff`

](RS\_Diag\_04180, RS\_Diag\_04159)

**[SWS\_DM\_01769]{DRAFT} Reporting 'DTC setting switched off' to IdsM**  
 [In case a diagnostic service `ControlDTCSetting` with subfunction Subfunction `0x02 (OFF)` and optionally with `DTCSettingControlOptionRecord` of value `0xFFFFFFFF` is received, the DM shall report a security event `SEV_UDS_DTC_SETTING_SWITCHED_OFF` to IdsM (see [11] and table [SWS\_DM\_01760]) with the context data given in [SWS\_DM\_01949].] (*RS\_Ids\_00810*)

**[SWS\_DM\_01949]{DRAFT} [**

Context Data:	-
---------------	---

**Table 7.22: SEV on 'DTC setting switched off'**

**]**(

**[SWS\_DM\_00811]{DRAFT} Re-enabling of ControlDTCSetting by Diagnostic Application** [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`

**]**(*RS\_Diag\_04159, RS\_Diag\_04211, RS\_Diag\_04067*)

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]{DRAFT} Consecutive registration of notifier with SetControlDtcStatusNotifier()** [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.] (*RS\_Diag\_04159*)

#### 7.4.2.8.23 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`).

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

**Table 7.23: Supported sub function of ResponseOnEvent (0x86)**

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]{DRAFT} Realisation of UDS service 0x86 ResponseOnEvent**  
 [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.] ([RS\\_Diag\\_04160](#))

**[SWS\_DM\_00493]{DRAFT} Reestablishing of Client Server communication**  
 [In case of a canceled diagnostic conversation this client receives the serviceToRespondTo-responses after a successful reestablishing of a diagnostic conversation.] ([RS\\_Diag\\_04160](#))

**[SWS\_DM\_00494]{DRAFT} Supported sub functions of ResponseOnEvent service** [The *Diagnostic Server instance* shall support the sub functions of diagnostic service 0x86 ResponseOnEvent as listed in Table 7.23 Supported sub function of Response on Event (0x86).] (*RS\_Diag\_04160*)

**[SWS\_DM\_01098]{DRAFT} Starting ResponseOnEvent in single and multiple client scenarios** [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.] (*RS\_Diag\_04160*)

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]{DRAFT} Support of eventWindowTime values for ResponseOnEvent** [The *Diagnostic Server instance* shall support the event window times as configured by the value of the DEXT parameter *DiagnosticResponseOnEvent.eventWindow.eventWindowTime*.] (*RS\_Diag\_04160*)

**[SWS\_DM\_01118]{DRAFT} Support of DEXT parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate** [The *Diagnostic Server instance* shall map the DEXT parameter *DiagnosticResponseOnEventClass.responseOnEventSchedulerRate* to the ISO 14229-1[1] parameter *responseOnEventSchedulerRate*.] (*RS\_Diag\_04160*)

**[SWS\_DM\_01119]{DRAFT} Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents** [The *Diagnostic Server instance* shall map the DEXT parameter *DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents* to the ISO 14229-1[1] parameter *maxNumChangeOfDataIdentifierEvents*.] (*RS\_Diag\_04160*)

**[SWS\_DM\_01120]{DRAFT} Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisonOfValueEvents** [The *Diagnostic Server instance* shall map the DEXT parameter *DiagnosticResponseOnEventClass.maxNumComparisonOfValueEvents* to the ISO 14229-1[1] parameter *MaxNumComparisonOfValueEvents*.] (*RS\_Diag\_04160*)

**[SWS\_DM\_01121]{DRAFT} Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength** [The *Diagnostic Server instance* shall map the DEXT parameter *DiagnosticResponseOnEventClass.maxSupportedDIDLength* to the ISO 14229-1[1] parameter *maxSupportedDIDLength*.] (*RS\_Diag\_04160*)

**[SWS\_DM\_01122]{DRAFT} Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumberOfStoredDTCStatusChangedEvents** [The *Diagnostic Server instance* shall map the DEXT parameter *DiagnosticResponseOnEventClass.maxNumberOfStoredDTCStatusChangedEvents* to the ISO 14229-1[1] parameter *maxNumberOfStoredDTCStatusChangedEvents*.] (*RS\_Diag\_04160*)



**[SWS\_DM\_01262]{DRAFT} No storage of RoE events** [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].] (*RS\_Diag\_04160*)

**[SWS\_DM\_01263]{DRAFT} Storage of RoE events** [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].] (*RS\_Diag\_04160*)

#### 7.4.2.8.24 Custom Diagnostic Services

In addition to supported UDS diagnostic services, as shown in [Table 7.7](#), 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]{DRAFT} Support for Custom Diagnostic Services** [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.] (*RS\_Diag\_04177*)

**[SWS\_DM\_00983]{DRAFT} Processing of Custom Diagnostic Services** [The `Diagnostic Server instance` shall call `ara::diag::GenericUDSService::HandleMessage` to process a custom UDS service by the given `SID`.] (*RS\_Diag\_04177*)

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 [ [10]].

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

#### 7.4.2.9 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]{DRAFT} Cancellation of a Diagnostic Conversation in case of External Service Processing** [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 [\[SWS\\_DM\\_00577\]](#).] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00278]{DRAFT} Cancellation of a Diagnostic Conversation in case of Internal Processing** [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.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00279]{DRAFT} Cancellation of a Diagnostic Conversation before Response Transmission** [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 `ara::diag::uds_transport::UdsTransportProtocolHandler::Transmit` of the respective UDS Transport Protocol Handler.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00280]{DRAFT} Cancellation of a Diagnostic Conversation at Response Transmission** [On Cancellation of a [Diagnostic Conversation](#) in case a diagnostic request is currently processed in context of this [Diagnostic Conversation](#) and `ara::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.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00847]{DRAFT} Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation** [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 [\[SWS\\_DM\\_00843\]](#).] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00577]{DRAFT} Canceling external service processors** [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`.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_00984]{DRAFT} Return of cancellation status** [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.] ([RS\\_Diag\\_04167](#))

**[SWS\_DM\_01348]{DRAFT} Consecutive registration of notifier with CancellationHandler::SetNotifier()** [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.] ([RS\\_Diag\\_04167](#))

### 7.4.3 Diagnostic SOVD Management

#### 7.4.3.1 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 7.4.2.1.1 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]{DRAFT} SOVD lock in UDS extended session** [Acquiring a SOVD lock shall only be possible if no UDS (ISO 14229-1) client is in extended session.] ([RS\\_Diag\\_04264](#))

**[SWS\_DM\_01476]{DRAFT} Parallel SOVD client handling** [The DM shall treat a SOVD client without a lock like a UDS (ISO 14229-1) client regarding parallel client handling.] ([RS\\_Diag\\_04264](#))

**[SWS\_DM\_01929]{DRAFT} SOVD Error for non-reentrant software** [If an SOVD resource is not available due to non-reentrant implementation, the DM shall return the error for "Conflicted state" as per [2].] ([RS\\_Diag\\_04264](#))

#### 7.4.3.2 SOVD Request Validation and Verification

##### 7.4.3.2.1 SOVD Authorization

**[SWS\_DM\_01472]{DRAFT} Redirection to authorization endpoint** [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.] ([RS\\_Diag\\_04268](#))

**[SWS\_DM\_01471]{DRAFT} Redirection to token endpoint** [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. *(RS\_Diag\_04268)*

#### 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]{DRAFT} Authorization validation** [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. *(RS\_Diag\_04268)*

**[SWS\_DM\_01469]{DRAFT} Validity period of authenticated roles** [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. *(RS\_Diag\_04268)*

**[SWS\_DM\_01781]{DRAFT} (Re-)Identification of Clients by Token** [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. *(RS\_Diag\_04268)*

**[SWS\_DM\_01782]{DRAFT} (Re-)Identification of Clients by Identity** [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. *(RS\_Diag\_04268)*

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.

#### 7.4.3.2.2 SOVD Lock

**[SWS\_DM\_01783]{DRAFT} Locking only for authorized SOVD clients** [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`. *(RS\_Diag\_04268)*

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

**[SWS\_DM\_01475]{DRAFT} DM shall allow only one lock per SOVD entity** [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.] (*RS\_Diag\_04264*)

**[SWS\_DM\_01474]{DRAFT} DM shall allow access only to unlocked SOVD entities** [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.] (*RS\_Diag\_04264*)

**[SWS\_DM\_01473]{DRAFT} DM shall lock SOVD entity after time expiration** [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.] (*RS\_Diag\_04264*)

#### 7.4.3.2.3 Validation of Environmental Conditions in SOVD

**[SWS\_DM\_01468]{DRAFT} Check of Environmental Conditions before executing SOVD methods** [Before executing a SOVD method, the Diagnostic Server instance shall evaluate the associated Environmental Conditions according to [SWS\_DM\_00111], [SWS\_DM\_00112], [SWS\_DM\_00286], [SWS\_DM\_00287] and [SWS\_DM\_00970].] (*RS\_Diag\_04256*)

**[SWS\_DM\_01467]{DRAFT} Environmental Condition Check for SOVD evaluated to TRUE** [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.] (*RS\_Diag\_04256*)

**[SWS\_DM\_01466]{DRAFT} Environmental Condition Check for SOVD evaluated to FALSE** [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.] (*RS\_Diag\_04256*)

#### 7.4.3.2.4 Service Validation of SOVD Requests

**[SWS\_DM\_01784]{DRAFT} Service Validation of SOVD Requests** [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]{DRAFT} Service Validation for SOVD Resource Collections** [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.]()

### 7.4.3.3 SOVD Data Conversion

The exchange of data from a SOVD Client to a SOVD Server and vice versa takes place on an interpreted physical level, whereas internally (as well as for UDS ISO 14220-1 [1]) the data is expressed by internal data types. To express this transformation `SwDataDefProps` were introduced in Diagnostic Extract Template[3]. This section describes how these transformations shall be interpreted for SOVD and how the subset of a JSON schema that represents a `DiagnosticDataElement` shall be derived based on Diagnostic Extract Template[3] so that the `Diagnostic Server` instance can offer consistent ASAM SOVD [2] APIs.

**[SWS\_DM\_01465]{DRAFT} SOVD data representation of atomic numeric data** [The atomic subset of a SOVD request or response body matching a `DiagnosticDataElement` with `baseTypeEncoding` NONE or 2C and a `compuMethod` of category IDENTICAL, LINEAR, SCALE\_LINEAR, RAT\_FUNC or SCALE\_RAT\_FUNC as defined in Diagnostic Extract Template[3] shall be of type `integer` or `number` and shall derive the JSON schema properties `type`, `minimum`, `maximum` and `multipleOf` based on `baseTypeEncoding`, `baseTypeSize`, `dataConstr` and `compuMethod` of the corresponding `SwDataDefProps`.](*RS\_Diag\_04256*)

**[SWS\_DM\_01464]{DRAFT} SOVD data representation of TEXTTABLE** [The atomic subset of a SOVD request or response body matching a `DiagnosticDataElement` with `baseTypeEncoding` NONE or 2C and a `compuMethod` of category TEXTTABLE as defined in Diagnostic Extract Template[3] shall be of type `string` with JSON property `enum`. For each `compuScale` (ordered) one entry of the `enum` property shall be derived from the `vt` of the `compuConst`.](*RS\_Diag\_04256*)

**[SWS\_DM\_01463]{DRAFT} SOVD data representation of atomic scaled numeric data with texttable** [The atomic subset of a SOVD request or response body matching a `DiagnosticDataElement` with `baseTypeEncoding` NONE or 2C and a `compuMethod` of category SCALE\_LINEAR\_AND\_TEXTTABLE, SCALE\_RAT\_AND\_TEXTTABLE as defined in Diagnostic Extract Template[3] shall be

of type [number, string] or [integer, string], shall derive the JSON schema parameter type, minimum, maximum and multipleOf based on baseTypeEncoding, baseTypeSize, dataConstr, numeric part of the compuMethod and shall derive the JSON schema parameter pattern from the texttable part of the compuMethod.] (RS\_Diag\_04256)

**[SWS\_DM\_01462]{DRAFT} SOVD data representation of TAB\_NOINTP** [The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category TAB\_NOINTP as defined in Diagnostic Extract Template[3] shall be of type number or integer with a enum property. For each CompuScale an entry of the enum property shall be derived from the vf of the CompuConst.] (RS\_Diag\_04256)

**[SWS\_DM\_01461]{DRAFT} SOVD data representation of BITFIELD\_TEXTTABLE** [The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding NONE or 2C and a compuMethod of category BITFIELD\_TEXTTABLE as defined in Diagnostic Extract Template[3] shall be of type object and shall derive one entry of type string with property enum per disjunct interval.] (RS\_Diag\_04256)

**[SWS\_DM\_01460]{DRAFT} SOVD data representation of baseTypeEncoding IEEE754** [The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding IEEE754 as defined in Diagnostic Extract Template[3] shall be of type number.] (RS\_Diag\_04256)

**[SWS\_DM\_01459]{DRAFT} SOVD data representation of textual Strings** [The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with baseTypeEncoding of ISO-8859-1, ISO-8859-2, WINDOWS-1252, UTF-8, UTF-16 or UCS-2 as defined in Diagnostic Extract Template[3] shall be of type string and the Diagnostic Server instance shall make the conversion based on the encodings to match the charset defined by ASAM SOVD[2].] (RS\_Diag\_04256)

**[SWS\_DM\_01458]{DRAFT} SOVD data representation of units** [The atomic subset of a SOVD request or response body matching a DiagnosticDataElement with a unit referenced as defined in Diagnostic Extract Template[3] shall include the unit in JSON schema according to ASAM SOVD.] (RS\_Diag\_04256)

**[SWS\_DM\_01457]{DRAFT} SOVD data representation of arrays** [The subset of a SOVD request and response body matching a DiagnosticDataElement fulfilling [TPS\_DEXT\_01001] shall be of type array with content type derived accordingly to the atomic data definition. The JSON property maxItems shall be derived from maxNumberOfElements of the DiagnosticDataElement.] (RS\_Diag\_04256)

#### 7.4.3.4 Standardized APIs

##### 7.4.3.4.1 Docs

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

**[SWS\_DM\_01456]{DRAFT} SOVD method Query an Online Capability Description** [Each [SOVD entity](#) shall support the [SOVD method Query an Online Capability Description](#) according to [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04257](#))

**[SWS\_DM\_01786]{DRAFT} SOVD Online Capability Description role sensitivity** [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](#).] ([RS\\_Diag\\_04257](#))

##### 7.4.3.4.2 Version-Info

The [SOVD API](#) uses [URI](#) based versioning. [SOVD](#) clients can determine which version is supported by an [SOVD](#) server by utilization of the resource `version-info` and querying the respective [URI](#) `https://<SOVD-Server-Host>/<OEM specific>/version-info`.

**[SWS\_DM\_01455]{DRAFT} SOVD method for SOVD API Versioning** [On [SOVD Server](#) level the [DM](#) shall support the [SOVD method](#) to access [SOVD version-info](#) according to [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01454]{DRAFT} Query to the SOVD API version** [Querying the [SOVD API version](#) shall only be supported by [SOVD entity SOVD Server](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01453]{DRAFT} path to version-info** [The path to the resource `version-info` shall remain the same for this and all future versions of the [SOVD API](#).] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01452]{DRAFT} Mismatching versions** [If an [SOVD](#) client uses a version not supported by the [SOVD](#) server, the server shall respond with [HTTP](#) status code `404-Not found`.] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01451]{DRAFT} SOVDInfo type** [The resource `version-info` shall be of type `SOVDInfo`.] ([RS\\_Diag\\_04256](#))

**[SWS\_DM\_01450]{DRAFT} SOVDInfo type content** [The type `SOVDInfo` shall contain attributes `base_uri` and `version`.] ([RS\\_Diag\\_04256](#))



#### 7.4.3.4.3 Data-categories

An [SOVD](#) client can read and write various kinds of data values from and to a [SOVD entity](#). Available standard `data-categories` used within data resource collections are identifications, measurements, parameters and system information.

**[SWS\_DM\_01449]{DRAFT} [SOVD method Retrieve Categories Supported by a Data Resource Collection](#)** [If a [SOVD entity](#) support [SOVD data](#) the [SOVD method Retrieve Categories Supported by a Data Resource Collection](#) shall be support according to [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04260](#), [RS\\_Diag\\_04258](#))

**[SWS\_DM\_01448]{DRAFT} [Standard resource Data categories](#)** [The standardized resource `Data categories` shall be supported by the entities [SOVD server](#), [Components](#).] ([RS\\_Diag\\_04260](#))

**[SWS\_DM\_01447]{DRAFT} [Data categories type](#)** [The resource `data-categories` shall be of type `DataCategory`.] ([RS\\_Diag\\_04260](#))

#### 7.4.3.4.4 Data-groups

`Data groups` are used to structure data in a specific way (e.g., engine performance related, ...) and are associated with a category.

**[SWS\_DM\_01446]{DRAFT} [SOVD method Retrieve Groups Supported by a Data Resource Collection](#)** [If a [SOVD entity](#) support [SOVD data](#) and if the [SOVD](#) has at least one [SOVD data-group](#) configured, the [SOVD method Retrieve Groups Supported by a Data Resource Collection](#) shall be supported according to [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04260](#), [RS\\_Diag\\_04258](#))

**[SWS\_DM\_01445]{DRAFT} [Data-groups type](#)** [The resource `data-groups` shall be of type `ValueGroup`.] ([RS\\_Diag\\_04260](#))

**[SWS\_DM\_01444]{DRAFT} [Data-groups type content](#)** [The `ValueGroup` type shall contain attributes `id` and `category`.] ([RS\\_Diag\\_04260](#))

#### 7.4.3.4.5 Locks

**[SWS\_DM\_01443]{DRAFT} [SOVD method for locking](#)** [The [DM](#) shall support [SOVD locks](#) on [Diagnostic Server instance](#) level according to [ASAM SOVD\[2\]](#).] ([RS\\_Diag\\_04264](#))

#### 7.4.3.4.6 Logging

[SWS\_DM\_01928]{DRAFT} **SOVD method for logging** [If a `DiagnosticSovdMethod` in context of a `DiagnosticSovdLog` is configured the corresponding Method for Logging shall be supported as per ASAM SOVD[2].] (*RS\_Diag\_04267*)

#### 7.4.3.5 Configurable APIs

This section describes how the `Diagnostic Server instance` realizes the configurable `SOVD APIs` as specified in ASAM SOVD[2]. Generally there are two different approaches that are used to realize these `SOVD APIs`.

For `SOVD APIs` that have a matching `UDS Services`, the realizing `ara::diag` interfaces are used for both protocols. Therefore this section aims on describing how the existing configuration patterns are interpreted to realize the corresponding `SOVD APIs`. For example a `DiagnosticDataIdentifier` with Service `ReadByIdentifier (0x22)` as of ISO 14229-1[1] is also available as `SOVD data` that supports the `SOVD Read Single Data Value from an Entity API`. This means that the underlying `ara::diag` interfaces will be used for both protocols. Thus the realizing application will only provide one `ara::diag` interface, that realizes the `UDS Service ReadByIdentifier (0x22)` as well as the `SOVD Read Single Data Value from an Entity API`. The following `SOVD` methods are realized by this pattern:

- `SOVD data`
- `SOVD configuration (parameter only)`
- `SOVD faults`
- `SOVD operations`
- Predefined `SOVD mode for CommunicationControl and ControlDTCSetting`

Unique `SOVD APIs`, that have no matching `UDS Service`, are realized by explicit modeling and by using dedicated `ara::diag` interfaces. These use-cases are solely These `SOVD` methods are realized explicitly:

- `SOVD bulk-data`
- `SOVD configuration as parameter and configuration as bulk-data`
- `SOVD software update`
- `SOVD logs`

#### 7.4.3.5.1 Data

**[SWS\_DM\_01442]{DRAFT} SOVD method Retrieve List of All Data Provided by the Entity** [If at least one SOVD data is configured, the *Diagnostic Server instance* shall provide the SOVD method Retrieve List of All Data Provided by the Entity according to ASAM SOVD[2.].] (*RS\_Diag\_04260*, *RS\_Diag\_04258*)

**[SWS\_DM\_01441]{DRAFT} SOVD data attribute id** [The SOVD attribute id of a SOVD data shall be derived from the *shortName* of the corresponding *DiagnosticDataIdentifier*.] (*RS\_Diag\_04260*)

**[SWS\_DM\_01440]{DRAFT} SOVD data attribute name** [The SOVD attribute name of a SOVD data shall be derived from the *longName* of the corresponding *DiagnosticDataIdentifier*.] (*RS\_Diag\_04260*)

**[SWS\_DM\_01439]{DRAFT} SOVD data attribute category** [The SOVD attribute category of a SOVD data shall be modelled using a SDG on the *DiagnosticDataIdentifier* with semantics as defined in Listing 7.1.] (*RS\_Diag\_04260*)

```

<SDG-DEF>
  <SHORT-NAME>SovdExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>DiagnosticDataIdentifier</SHORT-NAME>
      <GID>sovd:data</GID>
      <EXTENDS-META-CLASS> DiagnosticDataIdentifier </EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>sovd_data_category</SHORT-NAME>
          <CATEGORY>STRING</CATEGORY>
          <GID>sovd:data_category</GID>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
<DIAGNOSTIC-DATA-IDENTIFIER>
  <SHORT-NAME>DID</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="sovd:data">
        <SD GID="sovd:data_category">Measurements</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</DIAGNOSTIC-DATA-IDENTIFIER>

```

</DIAGNOSTIC-DATA-IDENTIFIER>

**Listing 7.1: SDG class for category of SOVD data. Including SDG-CLASS definition and an example.**

[SWS\_DM\_01437]{DRAFT} **SOVD group category uniqueness** [The aggregated SOVD data within one SOVD group shall all be of the same SOVD category.](RS\_Diag\_04260)

[SWS\_DM\_01436]{DRAFT} **SOVD group attribute category** [The SOVD attribute category of a SOVD group shall be derived by the category of the SOVD data within the SOVD group.](RS\_Diag\_04260)

[SWS\_DM\_01435]{DRAFT} **SOVD group id** [The SOVD attribute id of a SOVD group shall be derived from the `shortName` of the corresponding collection.](RS\_Diag\_04260)

[SWS\_DM\_01434]{DRAFT} **SOVD data attribute data of internal structure** [The SOVD attribute data of a SOVD data shall be represented by a JSON object literal with one entry per `DiagnosticParameter` of the corresponding `DiagnosticDataIdentifier`. The key of each entry shall be derived from the `shortName` of the corresponding `DiagnosticDataElement`.](RS\_Diag\_04260)

[SWS\_DM\_01433]{DRAFT} **SOVD method Read Single Data Value from an Entity** [For each SOVD data with Service ReadDataByIdentifier (0x22) the Diagnostic Server instance shall provide the SOVD method Read Single Data Value from an Entity according to ASAM SOVD [2].](RS\_Diag\_04260)

[SWS\_DM\_01432]{DRAFT} **SOVD method Read Single Data Value from an Entity data query** [The SOVD attribute data in the response body of the SOVD method Read Single Data Value from an Entity shall be queried analog to Service ReadDataByIdentifier (0x22) of ISO 14229-1[1] using the same instance of the data querying mechanism.](RS\_Diag\_04260)

[SWS\_DM\_01431]{DRAFT} **SOVD method Write a Data Value to an Entity** [For each SOVD data with Service WriteDataByIdentifier (0x2E) the Diagnostic Server Instance shall provide the SOVD method Write Single Data Value from an Entity according to ASAM SOVD[2].](RS\_Diag\_04260)

[SWS\_DM\_01430]{DRAFT} **SOVD method Write a Data Value to an Entity data processing** [The SOVD attribute data in the request body of the SOVD method Write a Data Value to an Entity shall be processed analog to Service WriteDataByIdentifier (0x2E) of ISO 14229-1[1] using the same instance of the processing mechanism.](RS\_Diag\_04260)

#### 7.4.3.5.2 Configuration

The ASAM SOVD[2] standard generally supports two different approaches for the API Methods for Configuration:

- Configurations as parameter using MIME type application/json with defined Response Body structure.
- Configurations as bulk-data using bulk like MIME types such as application/octet-stream

For the realization of these SOVD API Methods for Configuration two different approaches have been realized:

- Configurations as parameter derived from UDS Diagnostic-DataIdentifier
- Configurations as parameter or Configurations as bulk-data realized explicitly by modelling and using the dedicated `ara::diag::SovdConfiguration::SovdConfiguration` Interface

**[SWS\_DM\_01429]{DRAFT} SOVD method Retrieve List of All Configurations Provided by the Entity** [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[2].] (*RS\_Diag\_04261, RS\_Diag\_04258*)

#### 7.4.3.5.2.1 Derived Configurations

**[SWS\_DM\_01428]{DRAFT} SOVD configuration attribute id** [The SOVD attribute id of a SOVD configuration shall be derived from the `shortName` of the corresponding `DiagnosticDataIdentifier`.] (*RS\_Diag\_04261*)

**[SWS\_DM\_01427]{DRAFT} SOVD configuration attribute name** [The SOVD attribute name of a SOVD configuration shall be derived from the `longName` of the corresponding `DiagnosticDataIdentifier`.] (*RS\_Diag\_04261*)

**[SWS\_DM\_01426]{DRAFT} SOVD configuration attribute type** [The SOVD attribute type of a SOVD configuration shall be a constant of type string with value `parameter`.] (*RS\_Diag\_04261*)

**[SWS\_DM\_01425]{DRAFT} SOVD configuration attribute version and content\_type** [The SOVD attribute `version` and `content_type` of a SOVD configuration shall not be supported.] (*RS\_Diag\_04261*)

**[SWS\_DM\_01424]{DRAFT} SOVD method Read Configuration as Parameters** [For each SOVD configuration with Service `ReadDataByIdentifier` (0x22) configured the Diagnostic Server instance shall provide the SOVD method Read Configuration as Parameters according to ASAM SOVD[2].] (*RS\_Diag\_04261*)

**[SWS\_DM\_01423]{DRAFT} SOVD method Read Configuration as Parameters data query** [The SOVD attribute `data` in the response body of the SOVD method

Read Configuration as Parameters shall be queried analog to Service Read-DataByIdentifier (0x22) of ISO 14229-1[1] using the same instance of the data querying mechanism.](RS\_Diag\_04261)

**[SWS\_DM\_01422]{DRAFT} SOVD method Write Configuration as Parameters** [For each SOVD data with Service WriteDataByIdentifier (0x2E) configured the Diagnostic Server instance shall provide the SOVD method Write Configuration as Parameters according to ASAM SOVD[2].](RS\_Diag\_04261)

**[SWS\_DM\_01421]{DRAFT} SOVD method Write Configuration as Parameters data processing** [The SOVD attribute data in the request body of the SOVD method Write Configuration as Parameters shall be processed analog to Service WriteDataByIdentifier (0x2E) of ISO 14229-1[1] using the same instance of the processing mechanism.](RS\_Diag\_04261)

#### 7.4.3.5.2 Explicit Configurations

**[SWS\_DM\_01787]{DRAFT} Realization of SOVD Read Configuration** [If the SOVD method Read Configuration is called for an explicitly modelled SOVD Configuration the Diagnostic Server instance shall call `ara::diag::SovdConfiguration::RequestGetConfiguration.`](RS\_Diag\_04261)

**[SWS\_DM\_01788]{DRAFT} Realization of SOVD Write Configuration** [If the SOVD method Write Configuration is called for an explicitly modelled SOVD Configuration the Diagnostic Server instance shall call `ara::diag::SovdConfiguration::RequestPutConfiguration.`](RS\_Diag\_04261)

#### 7.4.3.5.3 Data-list

**[SWS\_DM\_01420]{DRAFT} SOVD data-list** [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[2].](RS\_Diag\_04260, RS\_Diag\_04258)

**[SWS\_DM\_01419]{DRAFT} SOVD method Read Multiple Data Values at Once from an Entity Using a Data List** [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.](RS\_Diag\_04260)

#### 7.4.3.5.4 Faults

**[SWS\_DM\_01418]{DRAFT} SOVD faults** [Each `DiagnosticTroubleCodeUds` shall express a `SOVD` fault.](*RS\_Diag\_04259*)

**[SWS\_DM\_01417]{DRAFT} SOVD method Read Faults from an Entity** [If at least one `SOVD` fault is configured the `Diagnostic Server` instance shall provide the `SOVD` method `Read Faults from an Entity` according to `ASAM SOVD`[2.](*RS\_Diag\_04259, RS\_Diag\_04258*)

**[SWS\_DM\_01416]{DRAFT} SOVD method Read Details for a Fault** [For each `SOVD` fault configured the `Diagnostic Server` instance shall provide the `SOVD` method `Read Details for a Fault` according to `ASAM SOVD`[2.](*RS\_Diag\_04259*)

**[SWS\_DM\_01415]{DRAFT} SOVD fault general attributes** [The `SOVD` attributes `code`, `scope`, `display_code`, `fault_name`, `severity` of a `SOVD` fault shall be derived according to [SWS\_DM\_01567].](*RS\_Diag\_04259*)

**[SWS\_DM\_01567]{DRAFT} SovdFaultAttributes** [

SOVD attribute	Deviation rules
code	<code>udsDtcValue</code> of <code>DiagnosticTroubleCodeUds</code>
scope	<code>shortName</code> of corresponding <code>DiagnosticMemoryDestination</code>
display_code	<code>shortName</code> of <code>DiagnosticTroubleCodeUds</code>
fault_name	<code>longName</code> of <code>DiagnosticTroubleCodeUds</code>
severity	<code>severity</code> of <code>DiagnosticTroubleCodeUds</code>

**Table 7.24: Response Body for `SOVD` fault attributes**

]()

**[SWS\_DM\_01414]{DRAFT} SOVD fault attribute status** [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`.](*RS\_Diag\_04259*)

**[SWS\_DM\_01413]{DRAFT} SOVD fault attribute symptom** [The `SOVD` attribute `symptom` of a `SOVD` fault shall be modelled using a `SDG` on the `DiagnosticTroubleCodeUds` with semantics as defined in Listing 7.2.](*RS\_Diag\_04259*)

```

<SDG-DEF>
  <SHORT-NAME>SovdExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>DiagnosticTroubleCodeUds</SHORT-NAME>
      <GID>sovd:fault_extensions</GID>
      <EXTENDS-META-CLASS>DiagnosticTroubleCodeUds</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>sovd_fault_symptom</SHORT-NAME>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
    
```

```

        <CATEGORY>STRING</CATEGORY>
        <GID>sovd:fault_symptom</GID>
    </SDG-PRIMITIVE-ATTRIBUTE>
</ATTRIBUTES>
</SDG-CLASS>
</SDG-CLASSES>
</SDG-DEF>
<DIAGNOSTIC-TROUBLE-CODE-UDS>
    <SHORT-NAME>DTC_UDS</SHORT-NAME>
    <ADMIN-DATA>
        <SDGS>
            <SDG GID="sovd:fault_extensions">
                <SD GID="sovd:fault_symptom">Some generic fault symptom
                    description.</SD>
            </SDG>
        </SDGS>
    </ADMIN-DATA>
</DIAGNOSTIC-TROUBLE-CODE-UDS>
    
```

**Listing 7.2: SDG class for symptoms of SOVD faults. Including SDG-CLASS definition and an example.**

**[SWS\_DM\_01412]{DRAFT} SOVD fault environment\_data** [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].] (*RS\_Diag\_04259*)

**[SWS\_DM\_01411]{DRAFT} SOVD fault environment\_data query** [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].] (*RS\_Diag\_04259*)

**[SWS\_DM\_01561]{DRAFT} Response Body for SOVD fault environment\_data table**

SOVD attribute / JSON object parameter key	SOVD Datatype	Deviation Rule
environment_data	object	Available if at least one <b>DiagnosticFreezeFrame</b> or <b>DiagnosticExtendedDataRecord</b> is configured for the <b>DTC</b>
environment_data / extended_data_records	object	Available if at least one <b>DiagnosticExtendedDataRecord</b> is configured for the <b>DTC</b>
environment_data / snapshots	object	Available if at least one <b>DiagnosticFreezeFrame</b> is configured for the <b>DTC</b>
environment_data / extended_data / {DiagnosticDataRecords.shortName}	object	The key { <b>DiagnosticDataRecords.shortName</b> } shall be derived from the <b>shortName</b> of the corresponding <b>DiagnosticDataRecords</b>
environment_data / extended_data / {DiagnosticDataRecords.shortName} / {DiagnosticDataElement.shortName}		The key { <b>DiagnosticDataElement.shortName</b> } shall be derived from the <b>shortName</b> of the corresponding <b>DiagnosticDataElement</b> . The object shall contain the <b>EDR</b> of the corresponding fault. The content shall be represented analog to the content of the <b>SOVD</b> method <b>Read Single Data Value</b> from an Entity.



SOVD attribute / JSON object parameter key	SOVD Datatype	Deviation Rule
environment_data / snapshots / {DiagnosticFreezeFrame.shortName}	object	The key {DiagnosticFreezeFrame.shortName} shall be derived from the shortName of the corresponding DiagnosticFreezeFrame.
environment_data / snapshots / {DiagnosticFreezeFrame.shortName} / id	string	The attribute id shall be derived from the shortName of the corresponding DiagnosticFreezeFrame.
environment_data / snapshots / {DiagnosticFreezeFrame.shortName} / record_number	integer	The attribute record_number shall be derived from the recordNumber of the corresponding DiagnosticFreezeFrame.
environment_data / snapshots / {DiagnosticFreezeFrame.shortName} / data	object	
environment_data / snapshots / {DiagnosticFreezeFrame.shortName} / data / {DiagnosticDataElement.shortName}		The key {DiagnosticDataElement.shortName} shall be derived from the shortName of the corresponding DiagnosticDataElement. The object shall contain the Diagnostic Freeze Frame of the corresponding fault. The content shall be represented analog to the content of the SOVD method Read Single Data Value from an Entity.

**Table 7.25: Response Body for SOVD fault environment\_data**

](RS\_Diag\_04259)

**[SWS\_DM\_01409]{DRAFT} SOVD method Delete All Faults of an Entity** [If at least one SOVD fault is configured the Diagnostic Server instance shall provide the SOVD method Delete All Faults of an Entity according to ASAM SOVD[2].](RS\_Diag\_04259)

**[SWS\_DM\_01408]{DRAFT} SOVD method Delete All Faults of an Entity with scope** [On request of SOVD method Delete All Faults of an Entity with query parameter scope the Diagnostic Server instance shall match the query parameter scope to the shortName of all MemoryDestinations. If the parameter matches a MemoryDestination, all DiagnosticTroubleCodeUds in the MemoryDestination shall be cleared analog to Service ClearDiagnosticInformation (0x14) of ISO 14229-1[1], else the processing shall return status code 400.](RS\_Diag\_04259)

**[SWS\_DM\_01407]{DRAFT} SOVD method Delete All Faults of an Entity without scope** [On request of SOVD method Delete All Faults of an Entity without the query parameter scope the Diagnostic Server instance shall clear all DiagnosticTroubleCodeUds analog to Service ClearDiagnosticInformation (0x14) of ISO 14229-1[1].](RS\_Diag\_04259)

**[SWS\_DM\_01406]{DRAFT} SOVD method Delete Single Fault of an Entity** [For each SOVD fault configured the Diagnostic Server instance shall provide the SOVD method Delete Single Fault of an Entity according to

ASAM SOVD[2] and clear the corresponding `DiagnosticTroubleCodeUds` analog to Service ClearDiagnosticInformation (0x14) of ISO 14229-1[1].] ([RS\\_Diag\\_04259](#))

#### 7.4.3.5.5 Operations

[SWS\_DM\_01405]{DRAFT} **SOVD operations** [Each `DiagnosticRoutine` shall express a SOVD operation.] ([RS\\_Diag\\_04262](#))

[SWS\_DM\_01404]{DRAFT} **SOVD method Retrieve List of All Available Operations from an Entity** [If at least one SOVD operation is configured the `Diagnostic Server instance` shall provide the SOVD method Retrieve List of All Available Operations from an Entity according to ASAM SOVD[2].] ([RS\\_Diag\\_04262](#), [RS\\_Diag\\_04258](#))

[SWS\_DM\_01403]{DRAFT} **SOVD method Get Details of a Single Operation** [For each SOVD operation configured the `Diagnostic Server instance` shall provide the SOVD method Get Details of a Single Operation according to ASAM SOVD[2].] ([RS\\_Diag\\_04262](#))

[SWS\_DM\_01402]{DRAFT} **SOVD operation attribute id** [The SOVD attribute `id` of a SOVD operation shall be derived from the `shortName` of the corresponding `DiagnosticRoutine`.] ([RS\\_Diag\\_04262](#))

[SWS\_DM\_01401]{DRAFT} **SOVD operation attribute name** [The SOVD attribute `name` of a SOVD operation shall be derived from the `longName` of the corresponding `DiagnosticRoutine`.] ([RS\\_Diag\\_04262](#))

[SWS\_DM\_01400]{DRAFT} **SOVD operation attribute proximity\_proof\_required** [The SOVD attribute `proximity_proof_required` of a SOVD operation shall be modelled using a SDG on the `DiagnosticRoutine` with semantics as defined in Listing 7.3.] ([RS\\_Diag\\_04262](#))

```
<SDG-DEF>
  <SHORT-NAME>SovdExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>DiagnosticRoutine</SHORT-NAME>
      <GID>sovd:operation_extensions</GID>
      <EXTENDS-META-CLASS>DiagnosticRoutine</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>sovd_operation_proximity_proof_required</SHORT-NAME>
          <CATEGORY>BOOLEAN</CATEGORY>
          <GID>sovd:operation_proximity_proof_required</GID>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
```

```

</SDG-DEF>
<DIAGNOSTIC-ROUTINE>
  <SHORT-NAME>Routine</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="sovd:operation_extensions">
        <SD GID="sovd:operation_proximity_proof_required">TRUE</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</DIAGNOSTIC-ROUTINE>
    
```

**Listing 7.3: SDG class for proximity\_proof\_required of SOVD operations. Including SDG-CLASS definition and an example.**

**[SWS\_DM\_01399]{DRAFT} SOVD operation attribute asynchronous\_execution** [If the `DiagnosticRoutine` only has a `DiagnosticStartRoutine` and no `DiagnosticStopRoutine` and no `DiagnosticRequestRoutineResults` configured, the attribute `asynchronous_execution` of the corresponding `SOVD` operation shall be `false`, else the attribute shall be `true`.] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01398]{DRAFT} SOVD operation proximity\_challenge** [If the `SOVD` attribute `proximity_proof_required` of a `SOVD` operation is `true` the `Diagnostic Server Instance` shall return the attribute `proximity_challenge` in the response body of `SOVD` method `Get Details of a Single Operation` using `ara::diag::SovdProximityChallenge::GetChallenge`, else the attribute `proximity_challenge` shall not be supported in the response body.] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01397]{DRAFT} SOVD operation mode** [The `SOVD` attribute `modes` of a `SOVD` operation shall not be supported.] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01396]{DRAFT} SOVD method Start Execution of an Operation** [For each `SOVD` operation configured the `Diagnostic Server Instance` shall provide the `SOVD` method `Start Execution of an Operation` according to [ASAM SOVD\[2\]](#). The `Diagnostic Server Instance` shall execute the method analog to `Service RoutineControl (0x31)` with `SubFunction Start (0x01)` of [ISO 14229-1\[1\]](#).] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01395]{DRAFT} SOVD method Start Execution of an Operation proximity\_response** [The `proximity_reponse` in the request body of `SOVD` method `Start Execution of an Operation` shall be validated using `ara::diag::SovdProximityChallenge::GetChallenge`.] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01394]{DRAFT} SOVD method Get Executions of an Operation** [For each `SOVD` operation supporting `asynchronous_execution` the `Diagnostic Server instance` shall support the `SOVD` method `Get Executions of an Operation` according to [SOVD SOVD\[2\]](#).] ([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01393]{DRAFT} SOVD method Get the Status of an Operation Execution** [For each `DiagnosticRoutine` with `DiagnosticRequestRoutineResults` the `SOVD Server Instance` shall provide the `SOVD` method `Get the`

Status of an Operation Execution according to [SOVD SOVD\[2\]](#). The [Diagnostic Server instance](#) shall execute the method analog to Service RoutineControl (0x31) with RequestResults (0x03) of ISO 14229-1[1] using the same instance of the processing mechanism.]([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01392]{DRAFT} SOVD method Stop the Execution of an Operation** [For each [DiagnosticRoutine](#) with [DiagnosticStopRoutine](#) the [SOVD Server Instance](#) shall provide the [SOVD method Stop the Execution of an Operation](#) according to [SOVD SOVD\[2\]](#). The [Diagnostic Server instance](#) shall execute the method analog to Service RoutineControl (0x31) with Stop (0x02) of ISO 14229-1[1] using the same instance of the processing mechanism.]([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01391]{DRAFT} SOVD method Stop the Execution of an Operation proximity\_response** [The `proximity_response` in the request body of [SOVD method Stop the Execution of an Operation](#) shall be validated using `ara::diag::SovdProximityChallenge::ValidateResponse`.]([RS\\_Diag\\_04262](#))

**[SWS\_DM\_01390]{DRAFT} SOVD operations request and response parameters** [The values for the key parameters in the request body and response body of the [SOVD methods Start Execution of an Operation, Stop the Execution of an Operation, Get the Status of an Operation Execution](#) shall be represented as [JSON object literal](#). The literal shall have one entry per [DiagnosticParameter](#) with key derived from the `shortName` of the corresponding [DiagnosticDataElement](#).]([RS\\_Diag\\_04262](#))

#### 7.4.3.5.6 Modes

**[SWS\_DM\_01389]{DRAFT} SOVD method Retrieve List of All Supported Modes of an Entity** [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\[2\]](#).]([RS\\_Diag\\_04263](#), [RS\\_Diag\\_04258](#))

#### 7.4.3.5.7 CommunicationControl

**[SWS\_DM\_01388]{DRAFT} SOVD mode commctrl** [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\[2\]](#).]([RS\\_Diag\\_04263](#))

**[SWS\_DM\_01387]{DRAFT} SOVD mode commctrl name** [The [SOVD attribute name](#) of the [SOVD mode commctrl](#) shall be a constant string with value `commctrl`.]([RS\\_Diag\\_04263](#))

**[SWS\_DM\_01386]{DRAFT} SOVD mode commctrl value schema** [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**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01385]{DRAFT} SOVD mode commctrl get value** [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 CommunicationControl (0x28)** of **ISO 14229-1[1]**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01384]{DRAFT} SOVD mode commctrl set value** [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.] (*RS\_Diag\_04263*)

#### 7.4.3.5.8 ControlDTCSetting

**[SWS\_DM\_01383]{DRAFT} SOVD mode dtcsetting** [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[2]**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01382]{DRAFT} SOVD mode dtcsetting name** [The **SOVD** attribute name of the **SOVD mode dtcsetting** shall be a constant string with value **dtcsetting**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01381]{DRAFT} SOVD mode dtcsetting value schema** [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]**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01380]{DRAFT} SOVD mode dtcsetting get value** [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]**.] (*RS\_Diag\_04263*)

**[SWS\_DM\_01379]{DRAFT} SOVD mode dtcsetting set value** [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]**.] (*RS\_Diag\_04263*)

#### 7.4.3.5.9 Bulk Data

This section describes realization of the [SOVD](#) API Methods for Handling of Bulk Data. This use-case will be mainly handled by the application to match the abstraction of different underlying file systems. [ara::diag::SovdBulkData::SovdBulkData](#) therefore mainly focuses on handling the underlying [SOVD](#) protocol. To allow for maximum flexibility during up- and download, [ara::diag::SovdBulkData::SovdBulkData](#) uses the data transfer strategies that have been introduced by [UDS Service 0x38 – RequestFileTransfer](#) and are described in detail in section [7.4.2.8.20](#).

**[SWS\_DM\_01789]{DRAFT} Realization of [SOVD](#) API Methods for Handling of Bulk Data** [The [Diagnostic Server instance](#) shall implement the [SOVD](#) API Methods for Handling of Bulk Data according to [ASAM SOVD \[2\]](#).] ([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01790]{DRAFT} Realization of [SOVD](#) Bulk Data API Method for Retrieve List of all Bulk Data Categories** [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 \[2\]](#).] ([RS\\_Diag\\_04266](#), [RS\\_Diag\\_04258](#))

**[SWS\_DM\_01791]{DRAFT} Realization of [SOVD](#) Read Bulk Data Meta Data** [If the [SOVD](#) method Read Bulk Data Meta Data is called the [Diagnostic Server instance](#) shall call [ara::diag::SovdBulkData::GetBulkDataMeta-Data](#).] ([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01792]{DRAFT} Realization of [SOVD](#) Download Bulk Data** [If the [SOVD](#) method Download Bulk Data is called the [Diagnostic Server instance](#) shall call [ara::diag::SovdBulkData::RequestBulkDataDownload](#).] ([RS\\_Diag\\_04266](#))

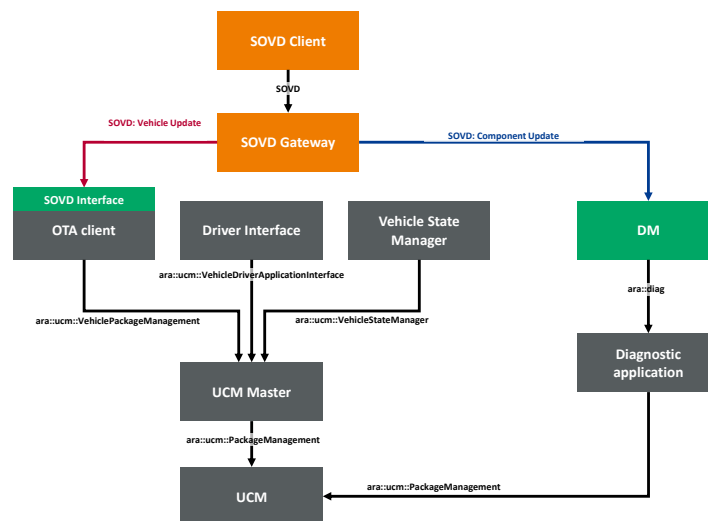
**[SWS\_DM\_01793]{DRAFT} Realization of [SOVD](#) Upload Bulk Data** [If the [SOVD](#) method Upload Bulk Data is called the [Diagnostic Server instance](#) shall call [ara::diag::SovdBulkData::RequestBulkDataUpload](#).] ([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01794]{DRAFT} Realization of [SOVD](#) Delete All Bulk Data Defined by Category** [If the [SOVD](#) method Delete All Bulk Data Defined by Category is called the [Diagnostic Server instance](#) shall call [ara::diag::SovdBulkData::DeleteAllBulkData](#).] ([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01795]{DRAFT} Realization of [SOVD](#) Delete Specific Bulk Data Resource** [If the [SOVD](#) method Delete Specific Bulk Data Resource is called the [Diagnostic Server instance](#) shall call [ara::diag::SovdBulkData::Delete-SpecificBulkData](#).] ([RS\\_Diag\\_04266](#))

### 7.4.3.5.10 Software Update

This section describes realization of the SOVD API Methods for Software Update. While the ASAM SOVD[2] 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 7.4.2.8.20).

**[SWS\_DM\_01796]{DRAFT} Realization of SOVD API Methods for Software Update** [The Diagnostic Server instance shall implement the SOVD API Methods for Software Update according to ASAM SOVD [2].] (RS\_Diag\_04265)

**[SWS\_DM\_01797]{DRAFT} Realization of SOVD Retrieve List of All Updates** [If the SOVD method Retrieve List of All Updates is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::GetAllUpdates`.] (RS\_Diag\_04265, RS\_Diag\_04258)

**[SWS\_DM\_01798]{DRAFT} Realization of SOVD Get Details of Update** [If the SOVD method Get Details of Update is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::GetUpdatePackageDetails.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01799]{DRAFT} Realization of SOVD Automated Installation of an Update** [If the SOVD method Automated Installation of an Update is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::PutUpdatePackageAutomated.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01800]{DRAFT} Realization of SOVD Prepare Installation of an Update** [If the SOVD method Prepare Installation of an Update is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::PrepareUpdatePackage.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01801]{DRAFT} Realization of SOVD Execute Installation of an Update** [If the SOVD method Execute Installation of an Update is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::ExecuteUpdatePackage.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01802]{DRAFT} Realization of SOVD Get Status of an Update** [If the SOVD method Get Status of an Update is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::GetUpdatePackageStatus.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01803]{DRAFT} Realization of SOVD Delete Update Package from an SOVD Server** [If the SOVD method Delete Update Package from an SOVD Server is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::DeleteUpdatePackage.`] (*RS\_Diag\_04265*)

**[SWS\_DM\_01804]{DRAFT} Realization of SOVD Register an Update at the SOVD Server** [If the SOVD method Register an Update at the SOVD Server is called the Diagnostic Server instance shall call `ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration.`] (*RS\_Diag\_04265*)

## 7.4.4 Diagnostic Event Management

### 7.4.4.1 Caching of application call for event processing

There is a difference between diagnostic service processing and diagnostic event management. Diagnostic services processing is always triggered by the DM as reaction of a received diagnostic request. For diagnostic event management the application is calling DM functions at any point in time and independent for a diagnostic service request. For the application calling such a function it is unknown if the DM is actually currently available. 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:

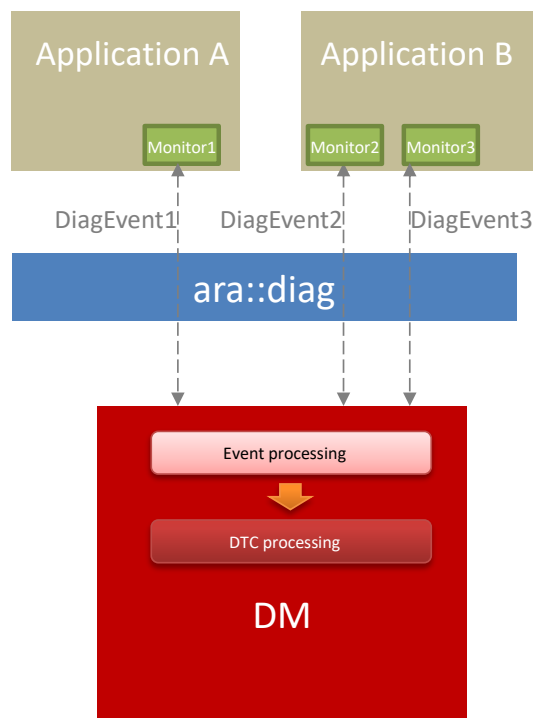
- Reporting states for enabled conditions
- Reporting state for clear `DTC` conditions
- Reporting state for monitor results

A detailed description of `APIs` that cache the function and also about what is cached, can be found in the individual chapters where these `APIs` are described.

## 7.4.4.2 Diagnostic Events

### 7.4.4.2.1 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]{DRAFT} MultipleEvent EventHandleType Generation** [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.

]([RS\\_Diag\\_04165](#), [RS\\_Diag\\_04201](#))

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]{DRAFT} Invalid EventHandleType** [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`.]([RS\\_Diag\\_04105](#))

#### 7.4.4.2.2 Monitors

A `diagnostic monitor` is a routine running inside an `AA` entity determining the proper functionality of a component. This monitoring function identifies a specific fault type (e.g. short-circuit to ground, missing signal, etc.) for a monitoring path. A monitoring path represents the physical system or a circuit, that is being monitored (e.g. sensor input). Each monitoring path is associated with exactly one `diagnostic event`.

In general `diagnostic monitors` are independent from the `DM`. Once the `ECU` is started and initialized they are permanently monitoring a part of the system and reporting the state to the `DM`. There are use cases, where it might not be required to continue to monitor the system part and the `monitor` could stop its task until a certain situation arises.

The `ara::diag::Monitor` interface is used by the `DM` to provide applications the ability to report monitor states to the `DM`. From this perspective the `ara::diag::Monitor` class is a required port. This class also has a notifier that is called by the

DM to inform the AA about monitor state changes. From this perspective the interface is a provided port. This makes the `ara::diag::Monitor` interface different to other classes and this is also the reason that `ara::diag::Monitor` has an `ara::diag::Monitor::Offer` and `ara::diag::Monitor::StopOffer` method that usually is only available for provided ports.

The `ara::diag::Monitor::Offer` method is necessary to provide an application the possibility to inform the DM that the application is ready to receive calls on the registered notifier. The notifier is registered in the constructor of the `ara::diag::Monitor` class. The concrete instance of the monitor object is only known after the object is created. As monitor class and the notifier instance are coupled (the notifier needs to know for which monitor it is used) the application needs to tell its notifier for which monitor object it is created. Therefore after creation of the `ara::diag::Monitor` class is not active until `ara::diag::Monitor::Offer` is called. Before `ara::diag::Monitor::Offer` is called, the application can complete the initialization of the notifier by providing the notifier implementation with a reference to the monitor object. Without this, the DM could call the notifier immediately after the object is constructed into a notifier object, that would not be fully initialized from application point of view.

The `ara::diag::Monitor::StopOffer` is the counterpart of the `ara::diag::Monitor::Offer`. At shutdown it defines a point in time where the application requests the monitor object to be no longer used and after `ara::diag::Monitor::StopOffer` it no longer expects calls on the notifier. After `ara::diag::Monitor::StopOffer` there are also no further calls on the corresponding `ara::diag::Event::GetFaultDetectionCounter` method. Without the `ara::diag::Monitor::StopOffer` it would be undefined until when an application will receive calls on the notifier and the notifier object could be safely discarded.

To avoid these kind of race conditions in object initialization/deinitialization the `ara::diag::Monitor::Offer` and `ara::diag::Monitor::StopOffer` methods are introduced. The names `ara::diag::Monitor::Offer` and `ara::diag::Monitor::StopOffer` are chosen to be the same names as the `ara::com Offer()` and `StopOffer()` as they have basically the same function. It would be of cause possible to name these methods like `Activate()` and `Deactivate()` but this would only introduce new names for exactly the same. Therefore AUTOSAR has chosen the `ara::diag::Monitor::Offer` and `ara::diag::Monitor::StopOffer` and it is emphasized that this is a `diag::Monitor` specific implementation not to be mixed with `ara::com Offer()` and `StopOffer()`.

Besides the reporting direction of the `monitors` (AAs report the monitoring status towards the DM), there is also a connection in the opposite direction: The DM uses the

- `initMonitor` (for Monitors with Monitor-internal debouncing)
- `initMonitor` (for Monitors with counter-based debouncing)
- `initMonitor` (for Monitors with time-based debouncing)

notifier to trigger a (re-)initialization of `diagnostic monitors` in the `AA`. The above listed notifier methods are optional and can be registered by one of the `ara::diag::Monitor` constructors.

**[SWS\_DM\_01740]{DRAFT} Initial enable condition state with notifier callback for disabled enable conditions** [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`.] (*RS\_Diag\_04270*)

**[SWS\_DM\_00562]{DRAFT} Monitor initialization for clearing reason** [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`.] (*RS\_Diag\_04185*)

**[SWS\_DM\_00563]{DRAFT} Monitor initialization for operation cycle restart reason** [If a `diagnostic event` was (re)started by calling the method `ara::diag::OperationCycle::RestartOperationCycle`, the `DM` shall call the registered notifier method `ara::diag::Monitor::Monitor::initMonitor` with the parameter `ara::diag::InitMonitorReason` set to `kRestart`.] (*RS\_Diag\_04186*)

**[SWS\_DM\_01094]{DRAFT} Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On** [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`.] (*RS\_Diag\_04125, RS\_Diag\_04159, RS\_Diag\_04270*)

**[SWS\_DM\_01095]{DRAFT} Monitor initialization for enable condition not fulfilled or ControlDTCSetting set to Off** [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`.] (*RS\_Diag\_04125, RS\_Diag\_04192, RS\_Diag\_04159*)

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]{DRAFT} Caching of monitor results** [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.] (*RS\_Diag\_04179*)

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]{DRAFT} MultipleMonitor MonitorHandleType Generation** [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.

]([RS\\_Diag\\_04063](#))

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]{DRAFT} Invalid MonitorHandleType** [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`.]([RS\\_Diag\\_04179](#), [RS\\_Diag\\_04063](#))

#### 7.4.4.2.3 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]{DRAFT} **Applicability of Event Combination** [The DM shall apply event combination for a DTC if more than one DiagnosticEventToTroubleCodeUdsMapping refer the same DiagnosticTroubleCodeUds.] (RS\_Diag\_04200)

Table 7.26 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.

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]	
....	....	....	....	....	....

**Table 7.26: Example of a configuration table including combined DTCs**

[SWS\_DM\_01107]{DRAFT} **DTC Status Byte calculation** [If DiagnosticCommonProps.typeOfEventCombinationSupported is configured, the DM shall calculate the UDS DTC status byte according to Table 7.27 for all events with event combination that are mapped to DTCs in that DiagnosticCommonProps.] (RS\_Diag\_04200)

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 Table 7.27 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 7.4.4.5.2 “UDS DTC Status”. The Derived event status is not available outside of the Diagnostic Manager.

UDS status bit description	Combined UDS status information logical equation
0 TestFailed	$CbDTC_{Bit0} = Event[1]_{Bit0} \mid Event[2]_{Bit0} \mid \dots \mid Event[n]_{Bit0}$
1 TestFailedThisOperationCycle	$CbDTC_{Bit1} = Event[1]_{Bit1} \mid Event[2]_{Bit1} \mid \dots \mid Event[n]_{Bit1}$
2 PendingDTC	$CbDTC_{Bit2} = Event[1]_{Bit2} \mid Event[2]_{Bit2} \mid \dots \mid Event[n]_{Bit2}$
3 ConfirmedDTC	$CbDTC_{Bit3} = Event[1]_{Bit3} \mid Event[2]_{Bit3} \mid \dots \mid Event[n]_{Bit3}$
4 TestNotCompletedSinceLastClear	$CbDTC_{Bit4} = ( Event[1]_{Bit4} \mid Event[2]_{Bit4} \mid \dots \mid Event[n]_{Bit4} ) \& ! CbDTC_{Bit5}$
5 TestFailedSinceLastClear	$CbDTC_{Bit5} = Event[1]_{Bit5} \mid Event[2]_{Bit5} \mid \dots \mid Event[n]_{Bit5}$
6 TestNotCompletedThisOperationCycle	$CbDTC_{Bit6} = ( Event[1]_{Bit6} \mid Event[2]_{Bit6} \mid \dots \mid Event[n]_{Bit6} ) \& ! CbDTC_{Bit1} CbDTC$



7	WarningIndicator Requested	$\text{CbDTC}_{\text{Bit}7} = \text{Event}[1]_{\text{Bit}7} \mid \text{Event}[2]_{\text{Bit}7} \mid \dots \mid \text{Event}[n]_{\text{Bit}7}$
---	----------------------------	---

**Table 7.27: Calculation of UDS status byte**

In table [Table 7.27](#) 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]{DRAFT} Clear all DTCs event with event combination** [If the [DM](#) is performing a clearDTC operation, the [DM](#) shall clear all related [events](#) independently if the event combination is used or not.] ([RS\\_Diag\\_04200](#))

**[SWS\_DM\_01109]{DRAFT} UDS DTC status update for combined DTCs** [Each time the status of an event is updated, the [DM](#) shall calculate the combined DTC status.] ([RS\\_Diag\\_04200](#))

**[SWS\_DM\_01110]{DRAFT} Callbacks for combined UDS DTC status change** [Each time the combined [DTC](#) status has changed, the [DM](#) shall call all configured callbacks.] ([RS\\_Diag\\_04200](#))

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]{DRAFT} Fault detection counter for combined events** [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.] ([RS\\_Diag\\_04200](#))

**[SWS\_DM\_01269]{DRAFT} Requesting DTC number for events without DTC** [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`.] ([RS\\_Diag\\_04200](#))

#### 7.4.4.2.3.1 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]{DRAFT} Event memory entry for events with the combination on storage** [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.]([RS\\_Diag\\_04200](#))

**[SWS\_DM\_01113]{DRAFT} Aging counter for combined events** [If `DiagnosticCommonProps.typeOfEventCombinationSupported` is set to `eventCombinationOnStorage`, the DM shall calculate the aging counter based on the combined DTC status.]([RS\\_Diag\\_04200](#))

#### 7.4.4.2.3.2 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]{DRAFT} Data storage for event combination on retrieval** [If `DiagnosticCommonProps.typeOfEventCombinationSupported` is set to `eventCombinationOnRetrieval`, the DM shall trigger the collection, update and storage of event related data per event.]([RS\\_Diag\\_04200](#), [RS\\_Diag\\_04067](#))

**[SWS\_DM\_01115]{DRAFT} Data reporting for event combination on retrieval** [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.]([RS\\_Diag\\_04200](#), [RS\\_Diag\\_04067](#))

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]{DRAFT} Reporting order of snapshot and extended data records** [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. | (*RS\_Diag\_04200*)

#### 7.4.4.2.4 EnableConditions

The `DM` provides also `Enable Conditions` to ignore a certain call of `ara::diag::Monitor::ReportMonitorAction` in required situations.

`DiagnosticEnableConditions` are mapped to `DiagnosticEvents` by `DiagnosticEventToEnableConditionGroupMappings`.

`Enable Conditions` are controlled via the function `ara::diag::Condition::SetCondition`. There current corresponding status, either `'kConditionFalse'` or `kConditionTrue` can be retrieved via `'ara::diag::Condition::GetCondition'`.

**[SWS\_DM\_00568]{DRAFT} Handling of enable conditions** [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. | (*RS\_Diag\_04192*)

Note: For a regular processing of `ara::diag::Monitor::ReportMonitorAction` **all** of the `enable conditions` mapped to the corresponding `event` have to be fulfilled.

The `DM` provides two types of interfaces that can be used from an `AA` to set or get conditionstates. The `ara::diag::Condition` interface provides operations that are assigned for one single condition, that is mapped via `DiagnosticEnableConditionPortMapping`. For each `DiagnosticCondition` the `AA` needs to instantiate exactly one `ara::diag::Condition` instance. The `ara::diag::MultipleCondition` provides the same functionality, but there is one interface that can be used for multiple `DiagnosticCondition` elements. This is helpful for applications that are handling many conditions and the creation of a large amount of `ara::diag::Condition` interfaces would be an overhead. In that case a single interface can be used for many conditions and each operation provides a further `conditionHandle` as parameter to identify the condition for which the operation is applicable. Each `DiagnosticCondition` requires a mapping via `DiagnosticMultipleConditionPortMapping` to express, that a multiple condition interface can be used for operation on this condition.

Function wise both interfaces offer the same kind of operations and do have the same semantics. For simplicity in this document the functionality of the condition class is described only via `ara::diag::Condition`. Whenever a functionality of the `ara::diag::Condition` class is described, the same applies for the operation on the `ara::diag::MultipleCondition`.

**[SWS\_DM\_01735]{DRAFT} MultipleCondition ConditionHandleType Generation**

[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]{DRAFT} Invalid ConditionHandleType** [If any `ara::diag::MultipleCondition` method with input parameter `conditionHandle` is called and this instance of `MultipleCondition` has not mapped `DiagnosticMultipleConditionPortMapping` that would create the provided `ConditionHandleType`, the DM shall return `kInvalidArgument`.]()

#### 7.4.4.2.5 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 (`FDCmax`) or -128 (`FDCmin`)), the `ara::diag::DebouncingState` is either `kFinallyDefective` or `kFinallyHealed`. This finally also leads than to a change of the `Event status` bit "kTestFailed".

There are two kinds of different debounce algorithms implemented by the DM:

- Counter-based debouncing ( see 7.4.4.2.5.1 "Counter-based debouncing")
- Time-based debouncing ( see 7.4.4.2.5.2 "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]). But since this functionality is not part of the DM, this document just describes this use case in 7.4.4.2.5.3 “Monitor-internal debouncing”).

If the FDC for an event with monitor-internal debouncing is needed, the registered function parameter `getFaultDetectionCounter` from the constructor `ara::diag::Monitor::Monitor` needs to be 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]{DRAFT} Debouncing parameters from DEXT** [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.] (*RS\_Diag\_04068*)

**[SWS\_DM\_01100]{DRAFT} Debouncing Algorithm not fitting** [If Debouncing parameters are present in DEXT and Debouncing Algorithm is not fitting (Timer or Counter based method is not configured appropriately), DM shall treat this as a Violation with the following error message: "Mismatching Debounce Algorithm in DEXT {modelledAlgorithm} and runtime {ctorAlgorithm}". {modelledAlgorithm} is the mapped debounce algorithm according to DiagnosticEventToDebounceAlgorithmMapping, {ctorAlgorithm} is the algorithm that is chosen in the ctor.] (*RS\_Diag\_04068*)

**[SWS\_DM\_01101]{DRAFT} Debouncing parameters from Monitor Constructor** [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.] (*RS\_Diag\_04068*)

#### 7.4.4.2.5.1 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]{DRAFT} Use of counter-based debouncing for events** [A `DiagnosticEvent` shall be subject to counter-based debouncing if the `DiagnosticEvent` is referenced in the role `diagnosticEvent` by a `DiagnosticEventToDebounceAlgorithmMapping`, where the referenced `debounceAlgorithm` aggregates a `DiagEventDebounceCounterBased` in the role `debounceAlgorithm`.] (*RS\_Diag\_04068*)

**[SWS\_DM\_00018]{DRAFT} Internal debounce counter init** [Upon start-up, the `DM` shall initialize the event's internal debounce counter to '0'.] (*RS\_Diag\_04068*)

**[SWS\_DM\_00017]{DRAFT} Calculation of the FDC based on the internal debounce counter** [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.] (*RS\_Diag\_04125, RS\_Diag\_04190*)

**[SWS\_DM\_00875]{DRAFT} Internal debounce counter incrementation** [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.] (*RS\_Diag\_04125, RS\_Diag\_04068*)

**[SWS\_DM\_00024]{DRAFT} Qualified failed event using counter-based debouncing** [If the internal debounce counter is greater or equal to `DiagEventDebounceCounterBased.counterFailedThreshold` the `DM` shall process the event as `kFinallyDefective`.] (*RS\_Diag\_04125, RS\_Diag\_04068*)

**[SWS\_DM\_00876]{DRAFT} Internal debounce counter decrementation** [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.] (*RS\_Diag\_04125, RS\_Diag\_04068*)

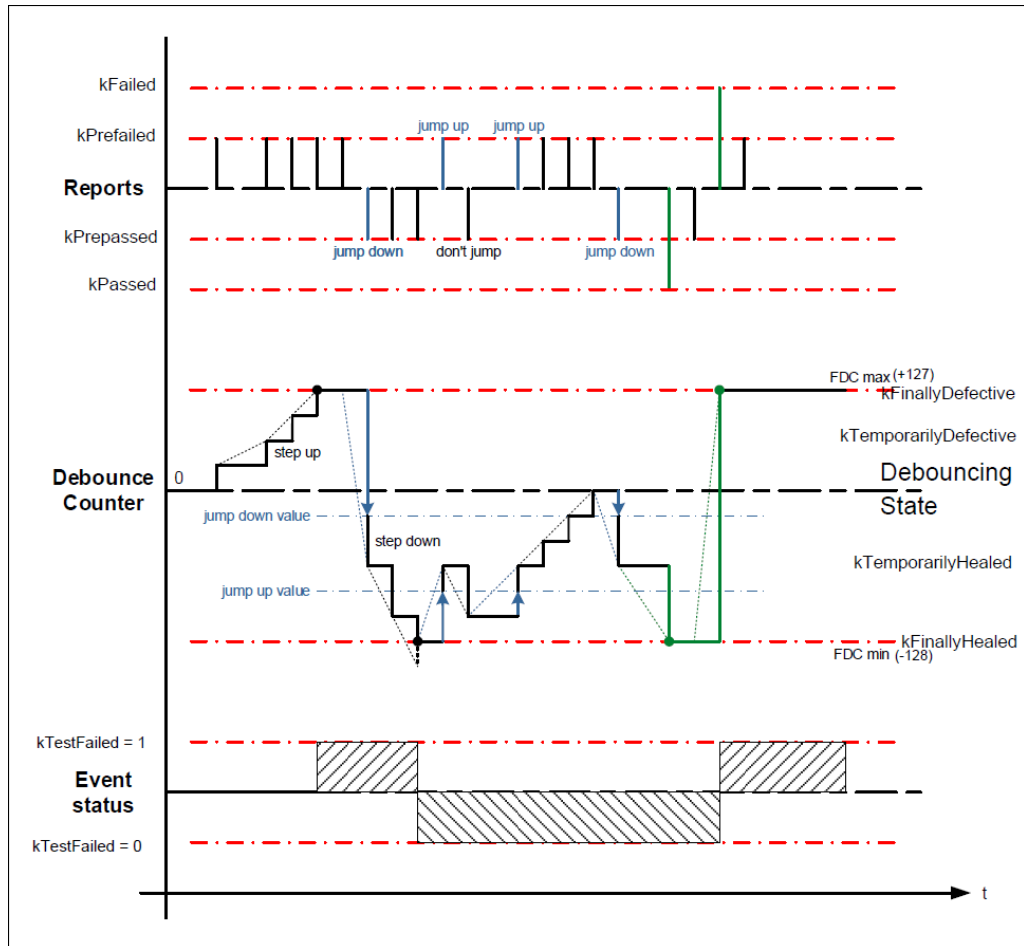
**[SWS\_DM\_00025]{DRAFT} Qualified passed event using counter-based debouncing** [If the internal debounce counter is less or equal to `DiagEventDebounceCounterBased.counterPassedThreshold` the `DM` shall process the event as `kFinallyHealed`.] (*RS\_Diag\_04125, RS\_Diag\_04068*)

**[SWS\_DM\_00021]{DRAFT} Direct failed qualification of counter-based events** [If the `monitor` reports `kFailed`, the DM shall set the internal debounce counter to the value `DiagEventDebounceCounterBased.counterFailedThreshold` and process the event as `kFinallyDefective`.] (*RS\_Diag\_04125*, *RS\_Diag\_04068*)

**[SWS\_DM\_00029]{DRAFT} Direct passed qualification of counter-based events** [If the `monitor` reports `kPassed`, the DM shall set the internal debounce counter to the value `DiagEventDebounceCounterBased.counterPassedThreshold` and process the event as `kFinallyHealed`.] (*RS\_Diag\_04125*, *RS\_Diag\_04068*)

**[SWS\_DM\_00022]{DRAFT} Internal debounce counter jump up behavior** [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.] (*RS\_Diag\_04068*)

**[SWS\_DM\_00023]{DRAFT} Internal debounce counter jump down behavior** [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.] (*RS\_Diag\_04068*)



**Figure 7.6: Counter-based debouncing**



#### 7.4.4.2.5.2 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]{DRAFT} Use of timer based debouncing for events** [The existence of a `DiagnosticEventToDebounceAlgorithmMapping` with an aggregation of `DiagEventDebounceTimeBased` by the referenced `DiagnosticDebounceAlgorithmProps.debounceAlgorithm` shall activate a time-based debouncing for this `event`.] (*RS\_Diag\_04225*)

**[SWS\_DM\_00085]{DRAFT} Internal debounce timer init** [The `DM` shall initialize the event's internal debounce timer to '0' upon start-up.] (*RS\_Diag\_04225*)

Note: `debounceCounterStorage` is not supported for time-based debouncing

**[SWS\_DM\_00030]{DRAFT} Calculation of the FDC based on the internal debounce timer** [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.] (*RS\_Diag\_04225, RS\_Diag\_04190*)

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]{DRAFT} Starting time-based event debouncing towards `kFinallyDefective`** [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`.

] (*RS\_Diag\_04225*)

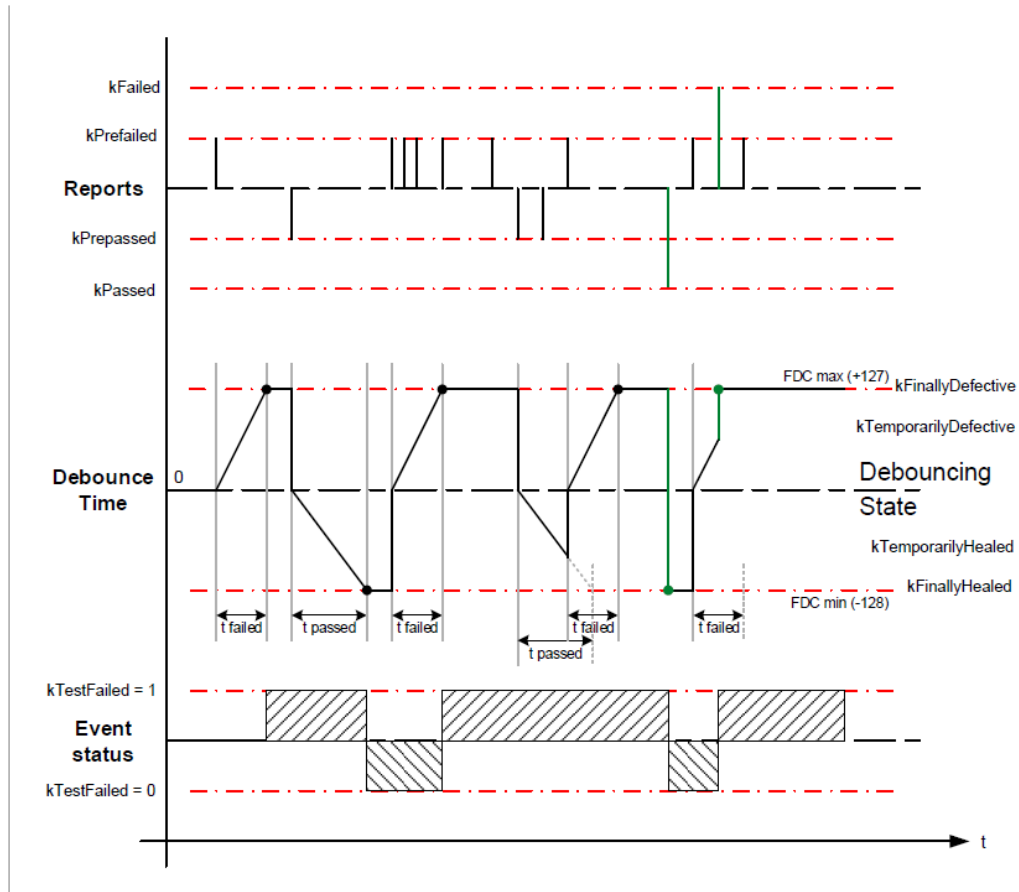
**[SWS\_DM\_00033]{DRAFT} Debounce timer behavior upon reported `kFailed`** [If the `monitor` reports `kFailed`, the `DM` shall set the debounce timer value to `DiagEventDebounceTimeBased.timeFailedThreshold` and process the event as `kFinallyDefective`.] (*RS\_Diag\_04225*)

**[SWS\_DM\_00878]{DRAFT} Starting time-based event debouncing towards `kFinallyHealed`** [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`.

](*RS\_Diag\_04225*)

**[SWS\_DM\_00036]{DRAFT} Debounce timer behavior upon reported `kPassed`** [If the `monitor` reports `kPassed`, the `DM` shall set the debounce timer value to `DiagEventDebounceTimeBased.timePassedThreshold` and process the event as `kFinallyHealed`.](*RS\_Diag\_04225*)



**Figure 7.7: Timer based debouncing**

**[SWS\_DM\_00880]{DRAFT} Debounce time freeze request** [If the `ara::diag::Monitor::ReportMonitorAction` method of a `ara::diag::Monitor` instance is called with the parameter `action` set to `kFreezeDebouncing`, for events with `DiagEventDebounceTimeBased` debouncing, the DM shall freeze the related debounce timer of the corresponding `event`.] (*RS\_Diag\_04068*, *RS\_Diag\_04225*)

Freezing of the timer is only supported for events with `DiagEventDebounceTimeBased` debouncing.

**[SWS\_DM\_00038]{DRAFT} Continuing a frozen debounce timer** [If a debounce timer is frozen (i.e. the corresponding `monitor` has called `ara::diag::Monitor::ReportMonitorAction` with the parameter `action` set to `kFreezeDebouncing`) and a new `kPrepassed` or `kPrefailed` is reported for this event, the DM module shall continue running the debounce timer starting with the frozen value.] (*RS\_Diag\_04225*)

#### 7.4.4.2.5.3 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.

**[SWS\_DM\_01267]{DRAFT} Reporting `kFdcThresholdReached` for monitor internal debouncing** [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].] (*RS\_Diag\_04127*)

**[SWS\_DM\_01268]{DRAFT} Value of `FaultDetectionCounter` in case of monitor internal debouncing** [The DM shall use the value returned in `ara::diag::Event::GetFaultDetectionCounter` as FDC of this event in case of internal debouncing, which means, that this event maps per `DiagnosticEventToDebounceAlgorithmMapping` to `DiagnosticDebounceAlgorithmProps.debounceAlgorithm`] (*RS\_Diag\_04068*)

#### 7.4.4.2.5.4 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]{DRAFT} Resetting counter-based debouncing** [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]{DRAFT} Resetting time-based debouncing** [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]{DRAFT} Resetting the internal debounce counter upon restarting an operation cycle** [If an operation cycle is restarted, the DM shall reset the internal debounce counter for all events referenced by `DiagnosticEventToOperationCycleMapping.diagnosticEvent` and referencing the restarted operation cycle by `DiagnosticEventToOperationCycleMapping.operationCycle`.] (*RS\_Diag\_04068, RS\_Diag\_04225*)

**[SWS\_DM\_00086]{DRAFT} Resetting the internal debounce counter after clearing DTC** [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.] (*RS\_Diag\_04068, RS\_Diag\_04225*)

#### 7.4.4.2.5.5 Dependencies to enable conditions

As described in section 7.4.4.2.4 `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\_00881]{DRAFT} Enable condition influence on debouncing behavior (freeze)** [If the `enable conditions` are not fulfilled for an `event` according to [SWS\_DM\_00568] and the debounce algorithm referenced by that event has the `DiagnosticDebounceAlgorithmProps.debounceBehavior` set to `freeze`, the DM shall freeze the according debounce timer or counter for the time the `enable conditions` are not fulfilled. This means that the debounce timer or counter remains unchanged.] (*RS\_Diag\_04192, RS\_Diag\_04125*)

**[SWS\_DM\_00882]{DRAFT} Enable condition influence on debouncing behavior (reset)** [If the `enable conditions` are not fulfilled for an `event` according to [SWS\_DM\_00568] and the debounce algorithm referenced by that event has the `DiagnosticDebounceAlgorithmProps.debounceBehavior` set to `reset`, the DM

shall reset the according internal debounce counter or timer and freeze it for the time the `enable conditions` are not fulfilled.]([RS\\_Diag\\_04192](#), [RS\\_Diag\\_04125](#))

#### 7.4.4.2.5.6 Dependencies to UDS service 0x85 ControlDTCSettings

**[SWS\_DM\_00088]{DRAFT} ControlDTCSetting influence (freeze)** [If ControlDTCSetting is set to disabled according to [\[SWS\\_DM\\_00910\]](#) and the debounce algorithm referenced has the `DiagnosticDebounceAlgorithmProps.debounceBehavior` set to `freeze`, the DM shall freeze the according internal debounce counter or timer for the time the ControlDTCSetting is set to disabled. This means that the internal debounce counter or timer remains unchanged.]([RS\\_Diag\\_04159](#), [RS\\_Diag\\_04125](#))

**[SWS\_DM\_00378]{DRAFT} ControlDTCSetting influence (reset)** [If ControlDTCSetting is set to disabled according to [\[SWS\\_DM\\_00910\]](#) and the debounce algorithm referenced has the `DiagnosticDebounceAlgorithmProps.debounceBehavior` set to `reset`, the DM shall reset the according internal debounce counter or timer and freeze it for the time the ControlDTCSetting is set to disabled.]([RS\\_Diag\\_04159](#), [RS\\_Diag\\_04125](#))

#### 7.4.4.2.6 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]{DRAFT} Event Status processing** [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.]([RS\\_Diag\\_04151](#))

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]{DRAFT} Event status bit transitions triggered by test results** [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.]([RS\\_Diag\\_04151](#))

Note that if debouncing for an `event` is configured, `kPrepassed` or `kPrefailed` reported via `ara::diag::Monitor::ReportMonitorAction`, trigger debounce mechanisms (see section [7.4.4.2.5](#)). 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]{DRAFT} Resetting the status of an Event** [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.]([RS\\_Diag\\_04151](#))

Rationale: This is an AUTOSAR-specific additional reset condition for the '`kTestFailed`' bit of the `Event status` bits.

**[SWS\_DM\_01027]{DRAFT} Event status bit transitions triggered by operation cycle restarting** [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`.]([RS\\_Diag\\_04178](#), [RS\\_Diag\\_04182](#))

**[SWS\_DM\_01028]{DRAFT} Event status bit transitions triggered by ClearDiagnosticInformation UDS service** [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;`

]([RS\\_Diag\\_04180](#))

#### 7.4.4.2.7 Event status change notifications

**[SWS\_DM\_00886]{DRAFT} Observability of the Event status byte** [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.]([RS\\_Diag\\_04183](#))

**[SWS\_DM\_01029]{DRAFT} Notification about Event status changes** [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`.]([RS\\_Diag\\_04183](#))

#### 7.4.4.2.8 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 data to that entry. The event occurrence counter is expected to have a size of one byte.

**[SWS\_DM\_00945]{DRAFT} Occurrence Counter initial value** [If a new `event memory` entry is created, the `DM` module shall initialize the associated occurrence counter with the value '1'.] ([RS\\_Diag\\_04067](#))

**[SWS\_DM\_00946]{DRAFT} Occurrence Counter increment strategy 'testFailed'-only** [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.] ([RS\\_Diag\\_04067](#))

**[SWS\_DM\_00947]{DRAFT} Occurrence Counter increment strategy 'confirmed-DtcBit'** [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.] ([RS\\_Diag\\_04105](#))

**[SWS\_DM\_00948]{DRAFT} Occurrence Counter upper limit** [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.] ([RS\\_Diag\\_04125](#))

A rollover of the occurrence counter byte from 255 to 0 must be avoided.

**[SWS\_DM\_01349]{DRAFT} Consecutive registration of notifier with SetEventStatusChangedNotifier()** [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.] ([RS\\_Diag\\_04125](#))

#### 7.4.4.3 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 [7.4.4.2.4 "EnableConditions"](#))
- `ClearDTCConditions` ( see [7.4.4.5.2 "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]{DRAFT} Caching of conditions** [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.] ([RS\\_Diag\\_04192](#))

#### 7.4.4.4 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]{DRAFT} Caching of RestartOperationCycle** [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.] ([RS\\_Diag\\_04178](#), [RS\\_Diag\\_04182](#))

**[SWS\_DM\_01104]{DRAFT} Operation Cycle restart** [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.] ([RS\\_Diag\\_04178](#))

**[SWS\_DM\_01105]{DRAFT} Restart OperationCycle during the processing of previous call** [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.] ([RS\\_Diag\\_04178](#))

**[SWS\_DM\_01358]{DRAFT} Consecutive registration of notifier with OperationCycle::SetNotifier()** [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.] ([RS\\_Diag\\_04178](#), [RS\\_Diag\\_04186](#))

#### 7.4.4.5 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]{DRAFT} Supported event memories** [The DM shall support the

- `primary event memory`
- up to 256 `user-defined event memories`

according to ISO 14229-1[1].] (*RS\_Diag\_04214, RS\_Diag\_04150*)

**[SWS\_DM\_00911]{DRAFT} Instances of DTCInformation interface** [The DM shall link each instance of the `ara::diag::DTCInformation` class with the mapped `DiagnosticMemoryDestination` referenced by the corresponding `DiagnosticMemoryDestinationPortMapping`.] (*RS\_Diag\_04214, RS\_Diag\_04150*)

**[SWS\_DM\_00056]{DRAFT} Availability of the primary event memory** [The DM shall support the `primary event memory` if a DTC exists having a `DiagnosticMemoryDestinationPrimary` referenced by its `DiagnosticTroubleCodeProps.diagnosticMemory`.] (*RS\_Diag\_04150*)

**[SWS\_DM\_00057]{DRAFT} Availability of a user-defined event memory** [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`.] (*RS\_Diag\_04214*)

The size of the different `event memories` is configurable by the DM configuration parameters.

**[SWS\_DM\_00920]{DRAFT} Configuration of the event memory size** [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`.] (*RS\_Diag\_04064*)

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.

#### 7.4.4.5.1 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]{DRAFT} Set of supported DTCs** [The existence of a *DiagnosticTroubleCodeUds* indicates that the DM shall support this DTC.] (*RS\_Diag\_04201*)

Note: Due to DM restrictions the 'DiagnosticTroubleCodeObd' and 'DiagnosticTroubleCodeJ1939' are not supported.

##### 7.4.4.5.1.1 Format

The DTC itself is a 3 byte value, that could have different interpretations.

**[SWS\_DM\_00058]{DRAFT} DTC interpretation format** [The DM shall use one internal DTC format interpretation that is defined in *DiagnosticMemoryDestinationPrimary.typeOfDtcSupported*.] (*RS\_Diag\_04157*)

Note: Refers to [TPS\_DEXT\_01008] in [3].

**[SWS\_DM\_CONSTR\_00059]{DRAFT} 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

] (*RS\_Diag\_04201*)

##### 7.4.4.5.1.2 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 `0xFFFFFFFF`.

**[SWS\_DM\_00064]{DRAFT} Definition of `DTC groups`** [The existence of a `DiagnosticTroubleCodeGroup` shall define the existence of the `DTC group` with the `DTC` identifier `DiagnosticTroubleCodeGroup.groupNumber`] (*RS\_Diag\_04117*, *RS\_Diag\_04115*)

Note: Refers to [TPS\_DEXT\_03014] in [3].

**[SWS\_DM\_00065]{DRAFT} Always supported availability of the group of all `DTCs`** [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`.] (*RS\_Diag\_04117*)

**[SWS\_DM\_CONSTR\_00082]{DRAFT} Restriction on the configuration of the `DTC group GroupOfAllDTCs`** [The `DM` shall ignore any configuration of a `DiagnosticTroubleCodeGroup.groupNumber` with a value of `0xFFFFFFFF`.] (*RS\_Diag\_04117*)

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.

#### 7.4.4.5.1.3 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) (*7.4.4.5.10*).

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]{DRAFT} Priority values** [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.] (*RS\_Diag\_04118*, *RS\_Diag\_04071*)

**[SWS\_DM\_CONSTR\_00961]{DRAFT} Limits of priority values** [The `DM` shall only support `DiagnosticTroubleCodeProps.priority` values in the range of 1..255.] (*RS\_Diag\_04118*, *RS\_Diag\_04071*)

## 7.4.4.5.2 UDS DTC Status

### 7.4.4.5.2.1 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 7.4.4.2.6. The *DM* provides means to other SW parts in order to control the *UDS DTC status bits*.

**[SWS\_DM\_00213]{DRAFT} DTC status processing** [The *DM* shall process the *UDS DTC status byte* harmonizing with the ISO 14229-1[1] standard.](*RS\_Diag\_04067*)

ISO 14229-1 Annex D generally defines *UDS DTC status byte* handling and the corresponding triggerings for them. The following requirements map interfaces and configuration parameters of the *DM* to generic *UDS DTC status bit* transition descriptions.

**[SWS\_DM\_00883]{DRAFT} UDS DTC status bit transitions triggered by test results** [The *DM* shall process the *UDS DTC status byte* based on the *Event status bits*.](*RS\_Diag\_04067*)

**[SWS\_DM\_00217]{DRAFT} UDS DTC status bit transitions triggered by ClearDiagnosticInformation UDS service** [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].](*RS\_Diag\_04180*, *RS\_Diag\_04067*)

**[SWS\_DM\_00218]{DRAFT} kConfirmedDTC (Bit3) calculation** [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*.](*RS\_Diag\_04136*, *RS\_Diag\_04067*, *RS\_Diag\_04151*)

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 7.4.4.5.2.4.

**[SWS\_DM\_01037]{DRAFT} Behavior of not configured DiagnosticEvent.confirmationThreshold** [If the optional parameter *DiagnosticEvent.confirmationThreshold* is not configured, the *DM* shall use a default value of '1' for that parameter.](*RS\_Diag\_04067*)

This means that a confirmedDTC is set to '1' along with a reported `kFailed` monitor `action` result.

#### 7.4.4.5.2.2 UDS DTC Status change notifications

**[SWS\_DM\_01030]{DRAFT} Observability of the UDS DTC status byte** [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.]([RS\\_Diag\\_04183](#))

**[SWS\_DM\_01031]{DRAFT} Notification about UDS DTC status changes** [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.]([RS\\_Diag\\_04183](#))

**[SWS\_DM\_01350]{DRAFT} Consecutive registration of notifier with SetDTCStatusChangedNotifier()** [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.]([RS\\_Diag\\_04183](#))

#### 7.4.4.5.2.3 Suppress DTC reporting

This chapter describes the dynamic suppression of DTC during runtime. A suppressed DTC is not visible for any tester and also not for internal AAs. The `Diagnostic Server instance` will not stop internal processing of a suppressed DTC. An use-case could be to hide a DTC in a certain market by an AA at run time.

**[SWS\_DM\_01308]{DRAFT} Functionality of SetDtcSuppression()** [If `ara::diag::DTCInformation::SetDtcSuppression` is called for a DTC and the UDS DTC status byte of that DTC is `pendingDTC = 0` and `confirmedDTC = 0`, the `Diagnostic Server instance` shall set the suppression status of the selected DTC to value of the parameter `suppressionStatus`.]([RS\\_Diag\\_04220](#))

**[SWS\_DM\_01307]{DRAFT} Precondition for suppression** [If `ara::diag::DTCInformation::SetDtcSuppression` with `suppressionStatus = kDTCSuppressionOn` is called for a DTC and the UDS DTC status byte of that DTC is `pendingDTC = 1` or `confirmedDTC = 1`, the suppression status shall remain unchanged and respond with `kSuppressionIgnored`.]([RS\\_Diag\\_04220](#))

**[SWS\_DM\_01306]{DRAFT} Functionality of GetDtcSuppression()** [If `ara::diag::DTCInformation::GetDtcSuppression` is called, the `Diagnostic`

`Diagnostic Server instance` shall return the `DTC` suppression status. A value of `kDTCsuppressionOn` means that the `DTC` is suppressed. A value of `kDTCsuppressionOff` means, that the `DTC` is currently not suppressed. *(RS\_Diag\_04220)*

`ara::diag::DTCInformation::GetDtcSuppression` and `ara::diag::DTCInformation::SetDtcSuppression` can be used on suppressed `DTCs` too.

**[SWS\_DM\_01305]{DRAFT} Behavior of services `ReadDTCInformation` with `DTC` mask record for suppressed `DTCs`** [In all sub functions of service `0x19 = ReadDTCInformation` with `DTC` mask record ( `0x01`, `0x02`, `0x07`, `0x0A` or `0x17`) the `Diagnostic Server instance` shall count or report only `DTCs` which are not suppressed.] *(RS\_Diag\_04220)*

**[SWS\_DM\_01304]{DRAFT} Behavior of services `ReadDTCInformation` for `ExtendedData` and `SnapshotData` for suppressed `DTCs`** [All sub functions of service `0x19 = ReadDTCInformation` with implicit `DTCs` status request ( `0x03`, `0x04`, `0x06`, `0x18` or `0x19` ) on a suppressed `DTC` the `Diagnostic Server instance` shall send a negative response (NRC `0x31`).] *(RS\_Diag\_04220)*

**[SWS\_DM\_01303]{DRAFT} Behavior of `Diagnostic Client` services `ClearDiagnosticInformation` for suppressed `DTCs`** [If the `Diagnostic Server instance` receives a `ClearDiagnosticInformation` (`0x14`) request with a `groupOfDTC` `DTC` `group` that is suppressed and different to any configured value of `DiagnosticTroubleCodeGroup.groupNumber` and has a value less than `0xFFFF00` (defined range for functional group identifier), the `Diagnostic Server instance` shall send a negative response with NRC `0x31` (`requestOutOfRange`).] *(RS\_Diag\_04220)*

**[SWS\_DM\_01302]{DRAFT} Behavior of `Diagnostic Client` services `ClearDiagnosticInformation` for suppressed `DTCs` inside a group** [Any request with a group of `DTCs` that is a `DTC` `group` (including `GroupOfAllDTCs`) containing a suppressed `DTC` shall clear this suppressed `DTC`.] *(RS\_Diag\_04220)*

There is no further influence of suppressed `DTCs` on the execution of service `ClearDiagnosticInformation` described in chapter 7.4.2.8.6 “`Service 0x14 – ClearDiagnosticInformation`”.

**[SWS\_DM\_01301]{DRAFT} Behavior of `ara::diag::Event` class methods for suppressed `DTCs`** [A suppressed `DTC` shall be treated like not configured for the interface `ara::diag::Event` with function `ara::diag::Event::GetDTCNumber` and shall return `kNoSuchDTC`.] *(RS\_Diag\_04220)*

All other functions of class `ara::diag::Event` are not affected by suppressed `DTCs`.

**[SWS\_DM\_01300]{DRAFT} Behavior of the `ara::diag::DTCInformation` class for suppressed `DTCs`** [A suppressed `DTC` shall be treated like not configured for the following functions of the class `ara::diag::DTCInformation`:

- `ara::diag::DTCInformation::GetCurrentStatus` shall return `kNoSuchDTC`.

- `ara::diag::DTCInformation::GetNumberOfStoredEntries` shall only count non suppressed DTCs.
- `ara::diag::DTCInformation::Clear` with parameter `dtcGroup`
  - is set to `DTC group` (including `GroupOfAllDTCs`) containing a suppressed DTC shall clear this suppressed DTC
  - which is suppressed and different to any configured value of `DiagnosticTroubleCodeGroup.groupNumber` and has a value less than `0xFFFF00` (defined range for functional group identifier), the DM shall trigger the error `kWrongDtc`.

](*RS\_Diag\_04220*)

**[SWS\_DM\_01299]{DRAFT} Enabling of the notification SetDTCStatusChangedNotifier() for suppressed DTCs** [If `ara::diag::DTCInformation::SetDTCStatusChangedNotifier` is called to enable the notification and the requested DTC is a suppressed DTC, the DM shall return `kNoSuchDTC`.] (*RS\_Diag\_04220*)

**[SWS\_DM\_01298]{DRAFT} Behavior of SetDTCStatusChangedNotifier() for suppressed DTCs** [If a notification on status change is enabled and is caused by a suppressed DTC this notification shall be suppressed.] (*RS\_Diag\_04220*)

**[SWS\_DM\_01297]{DRAFT} Behavior of SetNumberOfStoredEntriesNotifier() for suppressed DTCs** [If a notification on number of stored fault memory entries change is enabled, a suppressed DTC shall not release a notification triggered by suppressing or unsuppressing a DTC.] (*RS\_Diag\_04220*)

All other functions of interface `ara::diag::DTCInformation` are not affected by a suppressed DTC.

**[SWS\_DM\_01296]{DRAFT} Behavior of service ResponseOnEvent with subfunction onDTCStatusChange for suppressed DTCs** [Suppressed DTCs shall not lead to `serviceToRespondTo` messages for service `ResponseOnEvent(0x86)` with subfunction `onDTCStatusChange(0x01)`.] (*RS\_Diag\_04220*)

#### 7.4.4.5.2.4 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]{DRAFT} Observability of indicator status** [The DM shall provide the status of an indicator via the function `ara::diag::Indicator::GetIndicator` of the corresponding `ara::diag::Indicator` instance.] (*RS\_Diag\_04204*)



Note that the status of an indicator is determined by all the status information votes provided by events assigned to the corresponding indicator.

**[SWS\_DM\_00223]{DRAFT} Handling of 'warningIndicatorRequested' bit** [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.] ([RS\\_Diag\\_04204](#))

For confirmation check [[SWS\\_DM\\_00218](#)].

**[SWS\_DM\_01032]{DRAFT} Handling of 'WIR' bit without connected indicators** [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'.] ([RS\\_Diag\\_04204](#))

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]{DRAFT} Indicator healing** [If any indicator is configured, the `DM` shall set at the end of an operation cycle the UDS `DTC status bit 7` (`WarningIndicatorRequested`) to 0, if the following conditions are fulfilled:

- at least one `healingCycleCounterThreshold` is greater than 0
- all respective indicator healing cycle counters, which count the number of tested and passed healing `OperationCycles`, have reached their `healingCycleCounterThreshold`
- `WIRbit` is not enabled by calling the API `SetLatchedWIRStatus()`

] ([RS\\_Diag\\_04204](#))

**[SWS\_DM\_01582]{DRAFT} Healing counter increment** [If the `healingCycle` of the `DiagnosticConnectedIndicator` is restarted and `testFailedThisOperationCycle` bit is 0, the `DM` shall increment the healing counter by 1.] ([RS\\_Diag\\_04204](#))

**[SWS\_DM\_01266]{DRAFT} Warning Indicator Request Activation** [The `DM` shall set the `kWarningIndicatorRequested` of the `DTC` status always in the same time when the `kConfirmedDTC` bit is set to 1.] ([RS\\_Diag\\_04204](#))

This means that the `DiagnosticConnectedIndicator.indicatorFailureCycleCounterThreshold` is not evaluated by the `DM`.

**[SWS\_DM\_01359]{DRAFT} Consecutive registration of notifier with `Indicator::SetNotifier()`** [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.] ([RS\\_Diag\\_04204](#))

#### 7.4.4.5.2.5 User controlled WarningIndicatorRequest-bit

In some cases (e.g. controlling a failsafe reaction in an application) the WIR-bit (WarningIndicatorRequest-bit) of a corresponding event in DM shall be set/reset by a dedicated "fail-safe AA".

The "failsafe AA" has to ensure a proper status of the WIR-bit (e.g. regarding to ISO-14229-1[2] or manufacture specific requirements).

The failsafe AA shall report the required WIR-status to DM (via the function `ara::diag::Event::SetLatchedWIRStatus` of the corresponding `ara::diag::Event` instance) and has to ensure that the current WIR-status of an event (in DM) fits to the current fail-safe-status in the application:

- `fail-safe reaction` active: WIR-bit shall be set to "1"
- `fail-safe reaction` not active: WIR-bit shall be set to "0"

The fail-safe AA has to report the status after every change of its fail-safe state.

Therefore the DM provides the function `ara::diag::Event::SetLatchedWIRStatus` to set or reset the UDS DTC status bit "kWarningIndicatorRequested" of the related DTC.

Each invocation of the function `ara::diag::Event::SetLatchedWIRStatus` of an `ara::diag::Event` instance updates the WIR-bit for the corresponding DTC

Due to not storing the Status-Bit 7 ('warningIndicatorRequested' bit) on Shutdown, the fail-safe AA has to ensure that the 'warningIndicatorRequest' bit of a DTC fits to the current failsafe status after initialization of the DM.

Setting the WIR-bit of a DTC can be controlled via `SetLatchedWIRStatus()` OR by the DM internal WIR-bit handling. (OR-Operation).

**[SWS\_DM\_01033]{DRAFT} User controlled set of WIR-bit** [If the function `ara::diag::Event::SetLatchedWIRStatus` is called with parameter `status = TRUE`, the DM shall set the WIR-bit of the corresponding DTC to "1".] ([RS\\_Diag\\_04204](#))

**[SWS\_DM\_01034]{DRAFT} User controlled reset of WIR-bit** [If the function `ara::diag::Event::SetLatchedWIRStatus` is called with parameter `status = FALSE` and the DM internal WIR-bit handling is also not requesting it, the DM shall reset the WIR-bit of the corresponding event to "0".] ([RS\\_Diag\\_04204](#))

**[SWS\_DM\_01035]{DRAFT} User controlled WIR-bit handling and ControlDTCSetting** [If `ControlDTCSetting` is set to disabled according to [\[SWS\\_DM\\_00910\]](#) and the function `ara::diag::Event::SetLatchedWIRStatus` is called, the DM shall not change the status of the WIR-bit and the function shall return `kReportIgnored`.] ([RS\\_Diag\\_04204](#))

### 7.4.4.5.3 Destination

Each **DTC** is stored in one of the supported **event memories** according to [SWS\_DM\_00056] and [SWS\_DM\_00057].

**[SWS\_DM\_00083]{DRAFT} Event memory destination of an DTC** [The existence of `DiagnosticTroubleCodeProps.diagnosticMemory` shall assign all DTCs referencing this `DiagnosticTroubleCodeProps` to the **event memory** referenced by `DiagnosticTroubleCodeProps.diagnosticMemory`.] (*RS\_Diag\_04150, RS\_Diag\_04214*)

**[SWS\_DM\_CONSTR\_00084]{DRAFT} Each DTC shall be assigned to an event memory destination** [The **DM** shall only support **DTCs** with a configured `DiagnosticTroubleCodeProps.diagnosticMemory`.] (*RS\_Diag\_04150, RS\_Diag\_04214*)

### 7.4.4.5.4 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]{DRAFT} Persistent storage of event memory entries** [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)

] (*RS\_Diag\_04211, RS\_Diag\_04105*)

**[SWS\_DM\_00969]{DRAFT} Padding in case of failed data capturing** [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.] (*RS\_Diag\_04205*)

#### 7.4.4.5.4.1 Triggering for data storage

**[SWS\_DM\_00150]{DRAFT} Primary trigger for event memory entry storage** [Creating and storing memory entries (incl. collecting **DTC**-related data) shall be triggered according to the `DiagnosticMemoryDestination.memoryEntryStorageTrigger` configuration parameter (see [3]).] (*RS\_Diag\_04211, RS\_Diag\_04105*)

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]{DRAFT} Behavior of not configured `DiagnosticMemoryDestination.memoryEntryStorageTrigger`** [If the optional parameter `DiagnosticMemoryDestination.memoryEntryStorageTrigger` is not configured, the DM shall use a default value of 'testFailed' for that parameter.](/)

#### 7.4.4.5.4.2 Storage of `snapshot record` data

**[SWS\_DM\_00151]{DRAFT} `snapshot record` numeration** [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.]([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04189](#))

**[SWS\_DM\_00152]{DRAFT} Number of `snapshot records` for a DTC** [In case `DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration` is set to `calculated`, the number of snapshot record the DM is able to store for a DTC shall be determined by the `DiagnosticTroubleCodeProps.maxNumberFreezeFrameRecords` configuration parameter. In case `DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration` is set to `configured`, the number of `snapshot records` is determined by the number of `DiagnosticFreezeFrames` configured for a DTC.]([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04190](#))

Note that different `snapshot records` represent different snapshots collected in different points in time.

**[SWS\_DM\_01276]{DRAFT} Triggering for `snapshot record` storage (calculated, `maxNumberFreezeFrameRecords = 1`)** [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'.]([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04127](#))

**[SWS\_DM\_01277]{DRAFT} Triggering for `snapshot record` storage (calculated, `maxNumberFreezeFrameRecords > 1`)** [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`.]([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04127](#))

**[SWS\_DM\_01085]{DRAFT} Triggering for `snapshot record` storage (configured, without update)** [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`.] ([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04127](#))

**[SWS\_DM\_01086]{DRAFT} Triggering for snapshot record storage (configured, with update)** [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`.] ([RS\\_Diag\\_04205](#), [RS\\_Diag\\_04127](#))

**[SWS\_DM\_01087]{DRAFT} Snapshot record layout** [If any `snapshot record` storage trigger occurs, the DM shall capture the data defined by `DiagnosticTroubleCodeProps.snapshotRecordContent`. Each referenced `DiagnosticDataIdentifier` shall be captured via the `PortPrototype` configured for these DIDs.] ([RS\\_Diag\\_04205](#))

**[SWS\_DM\_00894]{DRAFT} Notification event upon snapshot record updates** [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.] ([RS\\_Diag\\_04148](#))

In case of

- deletion ([7.4.4.5.5](#))
- aging ([7.4.4.5.6](#))
- displacement ([7.4.4.5.10](#))

the DM doesn't trigger the notification calls for updates.

**[SWS\_DM\_01351]{DRAFT} Consecutive registration of notifier with SetSnapshotRecordUpdatedNotifier()** [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.] ([RS\\_Diag\\_04183](#))

#### 7.4.4.5.4.3 Storage of extended data

This section describes the configuration of and the access to extended data for a DTC.

**[SWS\_DM\_00154]{DRAFT} Number of extended data for a DTC** [The DM shall store zero or one extended data for a DTC. Extended data consists of `extended data records`. If at least one `DiagnosticTroubleCodeProps.extendedDataRecord` is configured for the corresponding DTC, the extended data shall be present in the event memory entry.] ([RS\\_Diag\\_04206](#), [RS\\_Diag\\_04190](#))

Note that contrary to `snapshot records`, `extended data records` do not necessarily represent data collected in different points in time. Extended data consists of a configurable number of `extended data records`, which are all collected when the respective memory entry is created in the event memory. The update mechanism of `extended data records` is configurable.

An `extended data record` typically contains `DM internal` data information. This is represented as `DiagnosticDataElement` referenced from `DiagnosticProvidedDataMapping`, where the `dataProvider` defines the content of the `internal` data. Such data elements can only be used within the scope of an `extended data records`. The DEXT limits a use in other `DiagnosticDataElement` such as from `snapshot records` or `DiagnosticDataElement` from `DIDs`.

**[SWS\_DM\_00155]{DRAFT} Extended data record numeration** [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`.] ([RS\\_Diag\\_04206](#), [RS\\_Diag\\_04189](#))

**[SWS\_DM\_00895]{DRAFT} Triggering for extended data record storage and updates** [The data collection and storage of the `extended data record` shall be triggered by the `DiagnosticExtendedDataRecord.trigger`. Updating extended data records after being first stored, shall be configurable with the `DiagnosticExtendedDataRecord.update` configuration parameter. The data layout of `extended data record` is defined by the order of `DiagnosticExtendedDataRecord.recordElement`. Each `DiagnosticDataElement` shall be captured in its order via the corresponding read function instance for Typed `DataElement`: `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortNameOfDEPortInterface::Read`.] ([RS\\_Diag\\_04206](#), [RS\\_Diag\\_04127](#))

#### 7.4.4.5.4.4 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 [Table 7.29](#) 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` ([7.4.2.8.7.5](#)).

The `internal data elements` with context "DEM" in [Table 7.29](#) are not additionally stored or frozen in the `extended data record` when the event memory storage is triggered.

**[SWS\_DM\_00949]{DRAFT} Generation and usage of internal DiagnosticDataElements** [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.]([RS\\_Diag\\_04127](#), [RS\\_Diag\\_04190](#))

**[SWS\_DM\_00950]{DRAFT} Configuration of DTC priority as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_PRIORITY`, the DM shall set the value of this `internal data element` to the DTC priority assigned by `DiagnosticTroubleCodeProps.priority` for this DTC. The length of this `internal data element` is one byte.]([RS\\_Diag\\_04190](#))

**[SWS\_DM\_00921]{DRAFT} Configuration of Error Memory Overflow Indication as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_OVFLIND`, the DM shall set the value of this `internal data element` to the DM-internal value for `event memory overflow` to which the related DTC belongs to and map it:

- "0" = False, in case no event memory overflow was detected.
- "1" = True, in case an event memory overflow was detected.

The length of this `internal DiagnosticDataElement` is one byte.]([RS\\_Diag\\_04093](#), [RS\\_Diag\\_04190](#))

For more details, see also [7.4.4.5.9 "Event memory overflow"](#).

**[SWS\_DM\_00951]{DRAFT} Configuration of DTC "current FDC" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_CURRENT_FDC`, the DM shall report the internal value of the current Fault Detection Counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord's DiagnosticDataElement`. The length of this internal data element is one byte.]([RS\\_Diag\\_04190](#))

The value translation from the internal debouncing mechanisms to the FDC is defined in [\[SWS\\_DM\\_00017\]](#) and [\[SWS\\_DM\\_00030\]](#).

**[SWS\_DM\_00952]{DRAFT} Configuration of DTC "max. FDC since clear" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_MAX_FDC_SINCE_LAST_CLEAR`, the DM shall report the internal value of the current maximum Fault Detection Counter since last clear of the contextual DTC in the respective `DiagnosticExtendedDataRecord's DiagnosticDataElement`. The length of this internal data element is one byte.]([RS\\_Diag\\_04068](#), [RS\\_Diag\\_04190](#))

**[SWS\_DM\_00953]{DRAFT} Configuration of DTC "max. FDC current cycle" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_MAX_FDC_DURING_CURRENT_CYCLE`, the DM shall

report the internal value of the current maximum Fault Detection Counter during the current operation cycle of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement`.

The length of this internal data element is one byte.]([RS\\_Diag\\_04127](#), [RS\\_Diag\\_04190](#))

**[SWS\_DM\_00954]{DRAFT} Configuration of DTC "occurrence counter" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_OCCCTR`, the DM shall report the internal value of the current occurrence counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement`.

The length of this internal data element is one byte.]([RS\\_Diag\\_04190](#))

For Event occurrence see [7.4.4.2.8](#).

**[SWS\_DM\_00955]{DRAFT} Configuration of DTC "aging counter up/down" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_AGINGCTR_UPCNT` or to `DEM_AGINGCTR_DOWNCNT`, the DM shall report the internal value of the current aging counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement` based on [\[SWS\\_DM\\_00956\]](#) or [\[SWS\\_DM\\_00957\]](#).]([RS\\_Diag\\_04190](#))

For Aging see [7.4.4.5.6](#).

**[SWS\_DM\_00956]{DRAFT} Configuration of DTC "aging counter up" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` is set to `DEM_AGINGCTR_UPCNT`, the DM shall map the internal aging counter in such a way that a counting-up mode from '0' to the `DiagnosticAging.threshold` value is created according to ISO 14229-1[1], Annex D.]([RS\\_Diag\\_04190](#))

**[SWS\_DM\_00957]{DRAFT} Configuration of DTC "aging counter down" as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` is set to `DEM_AGINGCTR_DOWNCNT`, the DM shall map the internal aging counter in such a way that a counting-down mode from the `DiagnosticAging.threshold` value to '0' is created.]([RS\\_Diag\\_04190](#))

**[SWS\_DM\_00958]{DRAFT} Default value for DTC "aging counter up" if aging is not allowed** [If the element `DiagnosticTroubleCodeProps.aging` does not exist and the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_AGINGCTR_UPCNT`, the DM shall set the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement` value to '0'.]([RS\\_Diag\\_04190](#))

**[SWS\_DM\_00959]{DRAFT} Default value for DTC "aging counter down" if aging is not allowed** [If the element `DiagnosticTroubleCodeProps.aging` does not exist and the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to



DEM\_AGINGCTR\_DOWNCNT, the DM shall set the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement` value to `DiagnosticAging.threshold` if configured or '255' otherwise. *(RS\_Diag\_04190)*

**[SWS\_DM\_CONSTR\_00960]{DRAFT} No support for DEM\_AGINGCTR\_UPCNT\_FIRST\_ACTIVE** ["DEM\_AGINGCTR\_UPCNT\_FIRST\_ACTIVE" for the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of a `DiagnosticParameter.dataElement` shall not be supported by the DM.] *(RS\_Diag\_04133)*

**[SWS\_DM\_00961]{DRAFT} Configuration of a DTCs significance as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_SIGNIFICANCE`, the DM shall set the value of this internal data element to the DTCs significance assigned by `DiagnosticTroubleCodeProps.significance` for this DTC and map it:

- "0" = occurrence.
- "1" = fault.

The length of this internal `DiagnosticDataElement` is one byte. *(RS\_Diag\_04190)*

**[SWS\_DM\_00962]{DRAFT} Configuration of a DTCs Failed Operation Cycles as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_FAILED_CYCLES`, the DM shall report the internal value of the current Failed Operation Cycles Counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement`. The length of this internal data element is one byte.] *(RS\_Diag\_04190)*

**[SWS\_DM\_00963]{DRAFT} Configuration of a DTCs failed operation Cycles Since First Failed as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_CYCLES_SINCE_FIRST_FAILED`, the DM shall report the internal value of the current Operation Cycles Since First Failed Counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement`. The length of this internal data element is one byte.] *(RS\_Diag\_04190)*

**[SWS\_DM\_00964]{DRAFT} Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record** [If the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` is set to `DEM_CYCLES_SINCE_LAST_FAILED`, the DM shall report the internal value of the current Operation Cycles Since Last Failed Counter of the contextual DTC in the respective `DiagnosticExtendedDataRecord`'s `DiagnosticDataElement`. The length of this internal data element is one byte.] *(RS\_Diag\_04190)*

#### 7.4.4.5.5 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]{DRAFT} Clearing a DTC group** [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.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00117]{DRAFT} Clearing a DTC** [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.] (*RS\_Diag\_04117*)

##### 7.4.4.5.5.1 Locking of the DTC clearing process by a client

The DM supports more than one Diagnostic Clients as described in section 7.4.2.1.1. All configured clients can simultaneously send a ClearDTC diagnostic request. This chapter describes the DM behavior in this situations.

**[SWS\_DM\_00144]{DRAFT} Parallel clearing DTCs in different DiagnosticMemoryDestination** [The DM shall support parallel clearing of DTCs if the target of the clear DTC operation is a different DiagnosticMemoryDestination.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00145]{DRAFT} Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination** [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.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00146]{DRAFT} Unlock clear DTC operation for one DiagnosticMemoryDestination** [After the DM has finished the clear DTC operation, it shall unlock the clear DTC operation for this DiagnosticMemoryDestination.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00147]{DRAFT} Behavior while trying to clear DTCs on a locked DiagnosticMemoryDestination** [If the DM is requested to clear DTCs of a DiagnosticMemoryDestination and the DM has locked this DiagnosticMemoryDestination for clearing DTCs according to [SWS\_DM\_00144], the DM shall refuse the second clear DTC operation and shall return a NRC 0x22 (ConditionsNotCorrect).] (*RS\_Diag\_04117*)

#### 7.4.4.5.5.2 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]{DRAFT} Handling of DiagnosticClearConditions** [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.] (*RS\_Diag\_04117*)

The effect of a forbidden clear DTC operation is described in the requirements below:

**[SWS\_DM\_00123]{DRAFT} Block clearing of UDS DTC status byte during a clear DTC operation** [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.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00124]{DRAFT} Limited clearing of UDS DTC status byte during a clear DTC operation** [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.] (*RS\_Diag\_04117*)

**[SWS\_DM\_00121]{DRAFT} Forbidden clearing of snapshot records and extended data records** [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.] (*RS\_Diag\_04117*)

#### 7.4.4.5.5.3 DTC clearing triggered by application

Besides the UDS request `ClearDiagnosticInformation` according to section 7.4.2.8.6.1 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]{DRAFT} Usage of ClearDTC Interface** [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.] (*RS\_Diag\_04194*)

**[SWS\_DM\_00898]{DRAFT} ClearDTC call on invalid `DTC` or `DTC group`** [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.] (*RS\_Diag\_04194*)

**[SWS\_DM\_00899]{DRAFT} ClearDTC called while another clear operation is in progress** [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`.] (*RS\_Diag\_04194*)

**[SWS\_DM\_00900]{DRAFT} ClearDTC processing in case of memory errors** [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`.] (*RS\_Diag\_04194*)

**[SWS\_DM\_00901]{DRAFT} Possible failure of ClearDTC** [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`.] (*RS\_Diag\_04194*)

#### 7.4.4.5.6 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]{DRAFT} Aging** [If the `DiagnosticTroubleCodeProps.aging` exists, The `DM` shall support `Aging` for all `DTCs` referring to this `DiagnosticTroubleCodeProps.aging`.] (*RS\_Diag\_04133*)

**[SWS\_DM\_00238]{DRAFT} Aging and healing** [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).] (RS\_Diag\_04133)

[SWS\_DM\_00239]{DRAFT} **Aging counter** [The DM shall support an Aging counter for each event memory entry.] (RS\_Diag\_04133)

Note that this counter shall be available as internal data element of extended data records.

The implementation of the internal aging counter mechanism is independent from the configuration parameter `DiagnosticProvidedDataMapping.dataProvider` of the corresponding `DiagnosticParameter.dataElement` set to `DEM_AGINGCTR_UPCNT` or to `DEM_AGINGCTR_DOWNCNT`. Only for reading the internal data element, a mapping as defined in [SWS\_DM\_00956] and [SWS\_DM\_00957] is applied.

[SWS\_DM\_00240]{DRAFT} **Processing the Aging counter** [The DM shall only allow processing the Aging counter if the related DTC is stored in the event memory, the status is qualified as passed ('testFailed' bit is set to 0) and healing, according to [SWS\_DM\_00238], is fulfilled.] (RS\_Diag\_04133)

[SWS\_DM\_00241]{DRAFT} **Aging cycle and threshold** [The Aging shall be calculated based on the referred `DiagnosticOperationCycle` via the reference `DiagnosticAging.agingCycle`.] (RS\_Diag\_04133)

[SWS\_DM\_01264]{DRAFT} **DiagnosticAging.threshold reached** [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.] (RS\_Diag\_04133)

[SWS\_DM\_01265]{DRAFT} **Aging requires tested cycles only** [If `DiagnosticMemoryDestination.agingRequiresTestedCycle` is set, the cycle shall only be considered in which the status was reported but not failed ('testNotCompletedThisOperationCycle' bit and 'testFailedThisOperationCycle' bit are set to 0).] (RS\_Diag\_04133)

[SWS\_DM\_00243]{DRAFT} **Aging-related UDS DTC status byte processing** [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`

] (RS\_Diag\_04140)

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]{DRAFT} **Behavior of not configured `DiagnosticMemoryDestination.agingRequiresTestedCycle`** [If the optional parameter `DiagnosticMemoryDestination.agingRequiresTestedCycle` is not configured, the DM shall use a default value of 'FALSE' for that parameter.]()

[SWS\_DM\_01577]{DRAFT} **Behavior of not configured `DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear`** [If the optional parameter `DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear` is not configured, the DM shall use a default value of 'statusBitNormal' for that parameter.]()

#### 7.4.4.5.7 NumberOfStoredEntries

[SWS\_DM\_00902]{DRAFT} **NumberOfStoredEntries** [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.](*RS\_Diag\_04109*)

[SWS\_DM\_01352]{DRAFT} **Consecutive registration of notifier with `SetNumberOfStoredEntriesNotifier()`** [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.](*RS\_Diag\_04109*)

#### 7.4.4.5.8 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'.

[SWS\_DM\_01575]{DRAFT} **Configuration of permanent storage of "TestFailed" status bits** [`DiagnosticMemoryDestination.statusBitStorageTestFailed` shall always be 'true'.]()

#### 7.4.4.5.9 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]{DRAFT} Persistent storage for event memory overflow information** [The `DM` module shall store and provide the `event memory overflow` information persistently for each of the configured `event memories` separately.] (*RS\_Diag\_04093*)

**[SWS\_DM\_00923]{DRAFT} Event memory overflow set condition** [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`.] (*RS\_Diag\_04093*)

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]{DRAFT} Event memory overflow reset condition** [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.] (*RS\_Diag\_04093*)

In case of `aging` and deleting single `DTCs`, the overflow indication of the `event memory` is not reset.

**[SWS\_DM\_00925]{DRAFT} Event memory overflow notifier on occurrence** [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`.] (*RS\_Diag\_04093*)

**[SWS\_DM\_00926]{DRAFT} Event memory overflow notifier on clear** [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`.] (*RS\_Diag\_04093*)

**[SWS\_DM\_01354]{DRAFT} Consecutive registration of notifier with SetEventMemoryOverflowNotifier()** [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.](*RS\_Diag\_04093*)

#### 7.4.4.5.10 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]{DRAFT} Disabled displacement** [If `DiagnosticMemoryDestination.eventDisplacementStrategy` selects `none` and an `overflow` situation occurred in that particular `event memory`, the `DM` shall discard the new reported event.](*RS\_Diag\_04118*)

**[SWS\_DM\_00928]{DRAFT} Priority and occurrence based displacement** [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](#).

]([RS\\_Diag\\_04118](#), [RS\\_Diag\\_04105](#))

For strategy [prioOcc](#) there is no [displacement](#) for equal or higher priority event memory entries. The UDS DTC status bits are also not considered.

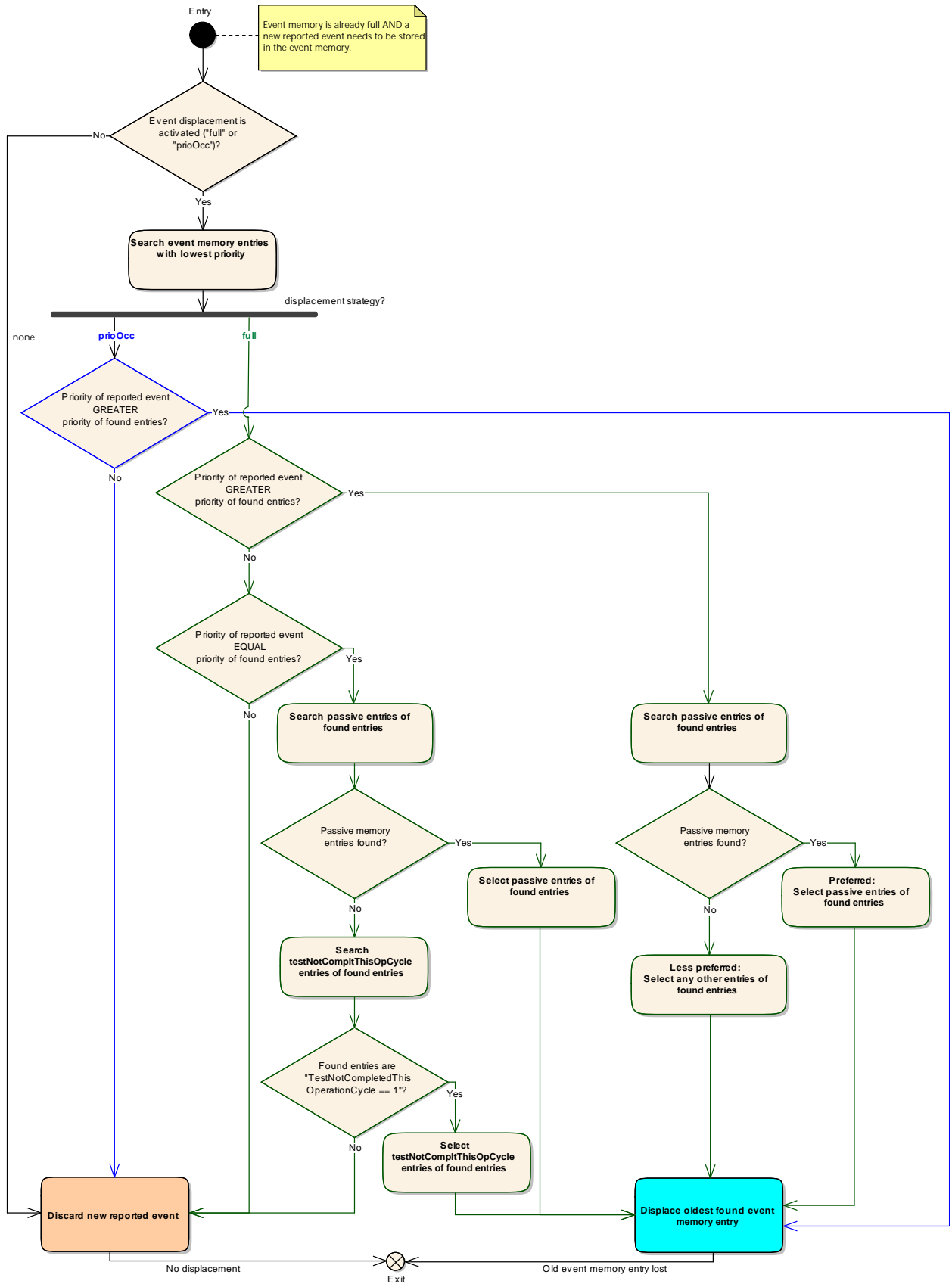
**[SWS\_DM\_00929]{DRAFT} Displacement strategy "full"** [If [DiagnosticMemory-Destination.eventDisplacementStrategy](#) selects [full](#) and an [overflow](#) situation occurred in that particular [event memory](#), the DM module shall perform the following selection sequence by combination of the different 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.

]([RS\\_Diag\\_04118](#), [RS\\_Diag\\_04105](#))

Details about Active / Passive Status are specified in [7.4.4.5.8 "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]{DRAFT} Displacement operation** [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.](RS\_Diag\_04118)

**[SWS\_DM\_01569]{DRAFT} Configurable reset of the pendingDTC bit in case of displacement** [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.](RS\_Diag\_04067, RS\_Diag\_04118)

**[SWS\_DM\_00932]{DRAFT} UDS DTC status bit 3 / 'ConfirmedDTC' after displacement** [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'.](RS\_Diag\_04067, RS\_Diag\_04118)

**[SWS\_DM\_00933]{DRAFT} UDS DTC status bit 5 / 'testFailedSinceLastClear' after displacement** [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'.](RS\_Diag\_04067, RS\_Diag\_04118)

**[SWS\_DM\_00934]{DRAFT} Condition for discarding the new event** [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*.](RS\_Diag\_04118)

#### 7.4.4.5.11 Reporting order of event memory entries

**[SWS\_DM\_00981]{DRAFT} Conditions of status based reporting order** [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.

](RS\_Diag\_04195)

**[SWS\_DM\_01566]{DRAFT} [**

SSID	Name
------	------

0x02	reportDTCByStatusMask(DTCStatusMask)
0x17	reportUserDefMemoryDTCByStatusMask(DTCStatusMask, MemorySelection)

**Table 7.28: Subfunctions of 0x19 / ReadDTCInformation with chronological reporting order**

]()

**[SWS\_DM\_00982]{DRAFT} Reporting order direction** [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.] ([RS\\_Diag\\_04195](#))

#### 7.4.5 Required Configuration

The Autosar Diagnostic Extract Template (DEXT) [\[3\]](#) is used for the DM configuration. By design this format is made as exchange format between the tools in the diagnostic workflow, in different steps data is added. To accommodate the fact that data is incomplete and refined in a later step, the DEXT [\[3\]](#) allows most of the elements to be optional and added at a later point in time. However at the point in time, when the DEXT [\[3\]](#) is used to configure the DM, a certain minimum content is required. In this chapter a loose list of DEXT [\[3\]](#) constraints is given. The mentioned elements need to be present so that the DM can be configured. Also the reaction on such missing elements is implementation specific, it is stated that the DM will not be able to behave as described in the document. A possible but not mandatory reaction is to refuse the DM generation at all and forcing the user to provide complete data.

**[SWS\_DM\_CONSTR\_00168]{DRAFT} Required operation cycles for diagnostic events** [Each `DiagnosticEvent` requires exactly one `DiagnosticEventToOperationCycleMapping` referencing the `diagnosticEvent` and one `DiagnosticOperationCycle`.] ([RS\\_Diag\\_04178](#))

**[SWS\_DM\_CONSTR\_00206]{DRAFT} Supported format for data identifier for VIN-DataIdentifier** [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`.] ([SRS\\_Eth\\_00026](#))

#### 7.4.6 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](#).

### 7.4.6.1 Internal and External Diagnostic Data Elements

A [DiagnosticDataElement](#) is called *internal* if there exists a [DiagnosticProvidedDataMapping](#) referencing this [DiagnosticDataElement](#), otherwise it is called an *external DiagnosticDataElement*.

Table 7.29 gives a list of the supported *internal DiagnosticDataElements*, where

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

**Content** describes the actual content of the data,

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

**Context** defines the exclusive context in which this [DiagnosticDataElement](#) is defined (if applicable). For "DEM" see [Diagnostic Event Management](#). For "DCM" see [Diagnostic Communication Management](#).

#### [SWS\_DM\_01565]{DRAFT} internaldataelements [

Data Provider	Content	Format	Context
DEM_AGINGCTR_DOWNCNT	Down-counting aging counter of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_AGINGCTR_UPCNT	Up-counting aging counter of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_CURRENT_FDC	Fault Detection Counter of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_CYCLES_SINCE_FIRST_FAILED	Operation Cycle Counter of contextual <a href="#">DTC</a> – Cycles since first failed	1 byte	DEM
DEM_CYCLES_SINCE_LAST_FAILED	Operation Cycle Counter of contextual <a href="#">DTC</a> – Cycles since last failed	1 byte	DEM
DEM_FAILED_CYCLES	Operation Cycle Counter of contextual <a href="#">DTC</a> – Failed cycles	1 byte	DEM
DEM_MAX_FDC_DURING_CURRENT_CYCLE	Fault Detection Counter maximum value during current operation cycle of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_MAX_FDC_SINCE_LAST_CLEAR	Fault Detection Counter maximum value since last clear of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_OCCCTR	Occurrence counter of contextual <a href="#">DTC</a>	1 byte	DEM
DEM_OVFLIND	Overflow indication of contextual <a href="#">DTCs</a> error memory(0 = False, 1 = True)	1 byte	DEM

Data Provider	Content	Format	Context
DEM_SIGNIFICANCE	Event significance of contextual <a href="#">DTC</a> (refer to <a href="#">DemDTCSignificance</a> ) (0 = OCCURRENCE, 1 = FAULT)	1 byte	DEM
DEM_PRIORITY	Priority of the contextual <a href="#">DTC</a>	1 byte	DEM
DCM_SESSION	Current session of contextual <a href="#">Diagnostic Conversation</a>	1 byte	DCM
DCM_SECURITY_LEVEL	Current security level of contextual <a href="#">Diagnostic Conversation</a>	1 byte	DCM
DEM_EVENT_ASSOCIATED_IDENTIFICATION	Represents the static value associated to it by <a href="#">associatedEventIdentification</a>	4 byte	DEM

**Table 7.29: Supported [internal DiagnosticDataElements](#)**

]()

**[SWS\_DM\_00393]{DRAFT} Retrieving data for [internal DiagnosticDataElements](#)** [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 [\[SWS\\_DM\\_01565\]](#).] ([RS\\_Diag\\_04097](#))

**[SWS\_DM\_CONSTR\_00394]{DRAFT} [Internal DiagnosticDataElements](#) are read-only** [A [DiagnosticDataIdentifier](#) referenced by a [DiagnosticWriteDataByIdentifier](#) service shall not contain any [internal DiagnosticDataElement](#).] ([RS\\_Diag\\_04097](#))

An [internal DiagnosticDataElement](#) is called DCM-exclusive resp. DEM-exclusive if the context of the name token described in [Table 7.29](#) 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 [Table 7.29](#) might in future be extended by [internal DiagnosticDataElements](#) not restricted to exclusive use within a DCM resp. DEM context.

**[SWS\_DM\_CONSTR\_00395]{DRAFT} 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](#).] ([RS\\_Diag\\_04097](#))

**[SWS\_DM\_CONSTR\_00396]{DRAFT} 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](#).] ([RS\\_Diag\\_04097](#))

Note: The notion of [internal](#) and [external](#) is exclusively defined for [DiagnosticDataElements](#) and does not apply to [DiagnosticDataIdentifier](#).

**[SWS\_DM\_00905]{DRAFT} Retrieving data for external DiagnosticDataElements** [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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface` class and call its `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Read` function.] (*RS\_Diag\_04097*)

Note: In general, there are multiple instances of `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface` 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 7.4.6.2.1.

**[SWS\_DM\_01275]{DRAFT} Namespace for typed DiagnosticDataElements interface** [If the typed `DiagnosticDataElements` interface is used, it shall fulfill [\[SWS\\_DM\\_00510\]](#) also.] (*RS\_Diag\_04097*)

## 7.4.6.2 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 7.4.6.1,
- 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.

### 7.4.6.2.1 Supported Diagnostic Mappings

Details regarding the modeling of diagnostic mappings can be found in the TPS Manifest Specification [10].

#### 7.4.6.2.2 Reading Diagnostic Data Identifier

**[SWS\_DM\_00401]{DRAFT} Reading Diagnostic Data Identifier on Data Element level** [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].] (*RS\_Diag\_04097*)

**[SWS\_DM\_00848]{DRAFT} Reading Diagnostic Data Identifier by typed DataIdentifier interface** [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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` class and associated to the `DiagnosticDataIdentifier` for reading the data.] (*RS\_Diag\_04097*)

**[SWS\_DM\_01038]{DRAFT} Reading Diagnostic Data Identifier by `ara::diag::GenericDataIdentifier` interface** [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 `ara::diag::GenericDataIdentifier` instance according to its `PortPrototype` mapping.] (*RS\_Diag\_04097*)

**[SWS\_DM\_00849]{DRAFT} Reading Diagnostic Data Identifier by `GenericUDSService` interface** [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`.] (*RS\_Diag\_04097*)

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.

**[SWS\_DM\_00850]{DRAFT} Default Service Interface for reading `DiagnosticDataIdentifier`** [If the `Diagnostic Server instance` is required to read data configured as `DiagnosticDataIdentifier` and none of the requirements [SWS\_DM\_00401], [SWS\_DM\_00848], [SWS\_DM\_00849] applies, then the `Diagnostic Server instance` shall utilize the associated `RPortPrototype` typed by the `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` and call its `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Read` function.] (*RS\_Diag\_04097*)



Note: The default configuration as described in [SWS\_DM\_00850] assumes, that there is a single instance of `PPortPrototype` defined in the system, matching the `RPortPrototype` associated to the requested `DiagnosticDataIdentifier`. In this case, it is part of integration step to link these two ports.

**[SWS\_DM\_01274]{DRAFT} Namespace for typed DiagnosticDataIdentifier interface** [If the typed `DiagnosticDataIdentifier` interface is used, it shall fulfill [SWS\_DM\_00510] also.](*RS\_Diag\_04097*)

#### 7.4.6.2.3 Writing Diagnostic Data Identifier

**[SWS\_DM\_00906]{DRAFT} Writing Diagnostic Data Identifier by DataIdentifier interface** [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.](*RS\_Diag\_04097*)

**[SWS\_DM\_01039]{DRAFT} Writing Diagnostic Data Identifier by typed DataIdentifier interface** [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 `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` according to its `PortPrototype` mapping.](*RS\_Diag\_04097*)

**[SWS\_DM\_00908]{DRAFT} Writing Diagnostic Data Identifier by GenericUDSService interface** [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`.](*RS\_Diag\_04097*)

**[SWS\_DM\_00907]{DRAFT} Default Service Interface for writing DiagnosticDataIdentifier** [If the `Diagnostic Server instance` is required to write data configured as `DiagnosticDataIdentifier` and none of the requirements [SWS\_DM\_00906], [SWS\_DM\_00908] applies, then the `Diagnostic Server instance` shall utilize the associated `RPortPrototype` typed by the `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface` and call `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Write`.](*RS\_Diag\_04097*)

Note: The default configuration as described in [SWS\_DM\_00907] assumes, that there is a single instance of `PPortPrototype` defined in the system matching the `RPortPrototype` associated to the requested `DiagnosticDataIdentifier`. In this case, it is part of integration step to link these two ports.

## 7.5 Security Events

### [SWS\_DM\_01760] Security events for Diagnostic Management [

Name	Description	ID
SEV_UDS_WRITE_INVALID_DATA	Tester writes invalid data with service WriteDataBy Identifier.	23
SEV_UDS_SECURITY_ACCESS_DENIED	Tester has send an diagnostic request without meeting the servers security level requirements for that service. NRC 0x33 (securityAccessDenied) was returned.	24
SEV_UDS_COMMUNICATION_CONTROL_SWITCHED_OFF	ECU communication was switched off with service CommunicationControl.	25
SEV_UDS_SERVICE_NOT_SUPPORTED	A diagnostic service war requested that is not supported or not supported in the active session. NRC 0x11 (serviceNotSupported) or NRC 0x7F (serviceNotSupportedInActiveSession) was returned.	26
SEV_UDS_SUBFUNCTION_NOT_SUPPORTED	A subfunction of a diagnostic service was requested that is not supported for the requested service.	27
SEV_UDS_INCORRECT_MESSAGE_LENGTH_OR_FORMAT	Received request message does not match the specified length or the format of the parameters is different from the specified ones for that service. NRC 0x13 (incorrectMessageLengthOrInvalidFormat) was returned.	28
SEV_UDS_REQUEST_SEQUENCE_ERROR	A diagnostic service was received in the wrong sequence order. NRC 0x24 (requestSequenceError) was returned.	29
SEV_UDS_REQUEST_OUT_OF_RANGE	A diagnsotic request with a parameter that is out range of the allowed bounds wa received. A NRC 0x31 (requestOutOfRange) was returned.	30
SEV_UDS_REQUESTED_ACTIONS_REQUIRES_AUTHENTICATION	A diagnostic request was received while the required authentication to executie this service is not given. NRC 0x34 (authenticationRequired) was returned.	31
SEV_UDS_SECURITY_ACCESS_NUMBER_OF_ATTEMPTS_EXCEEDED	Security Access failed and the number of invalid attempts has been exceeded. NRC 0x36 (exceedNumberOfAttempts) was returned.	32
SEV_UDS_SECURITY_ACCESS_INVALID_KEY	Security Access failed because the tester has send an invalid key to the server. An NRC 0x35 (invalidKey) was returned.	33
SEV_UDS_SECURITY_ACCESS_REQUIRED_TIME_DELAY_NOT_EXPIRED	A further attempt to unlock the ECU was taken before the delay time had expired. NRC 0x37 (requiredTimeDelayNotExpired) was sent.	34
SEV_UDS_NUMBER_OF_FAILED_AUTHENTICATION_ATTEMPTS_EXCEEDED	Number of failed authentication attempts - OEM specific	35
SEV_UDS_CERTIFICATE_FAILURE	The tester tried an authentication with an invalid certificate.	36
SEV_UDS_ECU_UNLOCK_SUCCESSFUL	Successful unlocked the ECU (via Security Access \$27), per Level	37
SEV_UDS_AUTHENTICATION_SUCCESSFUL	Succesfully authenticated (via Authentication \$29)	38
SEV_UDS_CLEAR_DTC_SUCCESSFUL	DTC information has been cleared by SID 0x14 Clear DTCInformation	39
SEV_UDS_ECU_RESET	ECU had been reset by SID 0x11 ResetECU	40



△

<b>Name</b>	<b>Description</b>	<b>ID</b>
SEV_UDS_WRITE_DATA	Diagnostic data identifier have been written by SID 0x2E.	41
SEV_UDS_REQUEST_DOWNLOAD	A diagnsotic data download sequence was requested with SID 0x34.	42
SEV_UDS_DTC_SETTING_SWITCHED_OFF	DTC setting switched off with service 0x85 (Control DTCSetting).	43

]([RS\\_Ids\\_00810](#))

## 8 API specification

This chapter lists all provided and required C++ API interfaces of the DM. The C++ API interfaces are divided into two parts:

- UDS Transportlayer interface  
A plug-in interface to extend the DM by own transport layers
- Diagnostic Application interface  
A `DiagnosticPortInterfaces` is representing a corresponding code instance. The deployment is simplified due to a direct mapping to DiagnosticObject in DEXT.

### 8.1 C++ language binding <sub component>

No content.

### 8.2 API Common Data Types

#### 8.2.1 C++ Diagnostic Error Types

**[SWS\_DM\_00544]{DRAFT} Use of general `ara::diag` errors** [Any Checked Error of a service interface shall be reported via the return type as specified in [6].] ([RS-AP\\_00128](#))

In `ara::diag`, there are the following types of Checked Errors:

1. Offer `ara::diag` errors: These errors can occur in a call of a any Offer interface method. They are defined in the error domain `ara::diag::DiagErrorDomain`.
2. Reporting `ara::diag` errors: These errors can occur in a call of a ReportMonitorAction interface method. They are defined in the error domain `ara::diag::DiagReportingErrorDomain`.
3. UDS NRC `ara::diag` errors: These errors can be returned by the skeletons. They are defined in the error domain `ara::diag::DiagUdsNrcErrorDomain`.
4. SOVD `ara::diag` errors: These errors can be returned by the skeletons for SOVD-specific error reporting. They are defined in the error domain `ara::diag::DiagSovdErrorDomain`.

Note: Beside the AUTOSAR defined error domains, platform vendors can optionally define their own vendor specific error domains.

**[SWS\_DM\_00514]{DRAFT} Definition of API enum ara::diag::DiagErrc [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DiagErrc	
<b>Underlying type:</b>	ara::core::ErrorDomain::CodeType	
<b>Syntax:</b>	enum class DiagErrc : ara::core::ErrorDomain::CodeType {...};	
<b>Values:</b>	kAlreadyOffered= 101	The service is already offered.
	kConfiguration Mismatch= 102	monitor configuration does not match dext
	kDebouncing Configuration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
	kReportIgnored= 104	Enable Conditions disabled, OC not started, ...
	kInvalidArgument= 105	e.g. kPreFailed with internal debouncing
	kNotOffered= 106	Offer not called before reporting.
	kNoSuchDTC= 108	No DTC available.
	kBusy= 109	Interface is busy with processing.
	kFailed= 110	Failed to process.
	kMemoryError= 111	A memory error occurred during processing.
	kWrongDtc= 112	A wrong DTC number was requested.
	kRejected= 113	Requested operation was rejected due to StateManagements/ machines internal state.
	kResetTypeNot Supported= 114	The requested Diagnostic reset type is not supported by the Diagnostic Address instance.
	kRequestFailed= 115	Diagnostic request could not be performed successfully.
kCustomResetTypeNot Supported= 116	The requested Diagnostic custom reset type is not supported by the Diagnostic Address instance.	
kSuppressionIgnored= 117	In case a suppression of a DTC is requested, but the conditions are not met.	
<b>Description:</b>	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00125](#))

**[SWS\_DM\_00515]{DRAFT} Definition of API class ara::diag::DiagException [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DiagException
<b>Base class:</b>	ara::core::Exception
<b>Syntax:</b>	class DiagException : public ara::core::Exception {...};
<b>Description:</b>	Exception type thrown by Diag classes.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00516]{DRAFT} Definition of API function ara::diag::DiagException::DiagException [**

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagException</a>
<b>Symbol:</b>	DiagException(ara::core::ErrorCode err)
<b>Syntax:</b>	explicit DiagException (ara::core::ErrorCode err) noexcept;
<b>Parameters (in):</b>	err   the ErrorCode
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Construct a new DiagException from an ErrorCode.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00517]{DRAFT} Definition of API class ara::diag::DiagErrorDomain [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DiagErrorDomain
<b>Base class:</b>	ara::core::ErrorDomain
<b>Syntax:</b>	class DiagErrorDomain final : public ara::core::ErrorDomain {...};
<b>Unique ID:</b>	0x8000'0000'0000'0401
<b>Description:</b>	Error domain for diagnostic errors.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00518]{DRAFT} Definition of API type ara::diag::DiagErrorDomain::Errc [**

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00519]{DRAFT} Definition of API type ara::diag::DiagErrorDomain::Exception [**

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00520]{DRAFT} Definition of API function ara::diag::DiagErrorDomain** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagErrorDomain</a>
<b>Symbol:</b>	DiagErrorDomain()
<b>Syntax:</b>	constexpr DiagErrorDomain () noexcept;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Default constructor.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00521]{DRAFT} Definition of API function ara::diag::DiagErrorDomain::Name** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagErrorDomain</a>
<b>Symbol:</b>	Name()
<b>Syntax:</b>	const char * Name () const noexcept override;
<b>Return value:</b>	const char * "Diag"
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00522]{DRAFT} Definition of API function ara::diag::DiagErrorDomain::Message** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagErrorDomain</a>
<b>Symbol:</b>	Message(ara::core::ErrorDomain::CodeType errorCode)
<b>Syntax:</b>	const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;
<b>Parameters (in):</b>	errorCode the error code value
<b>Return value:</b>	const char * the text message, never nullptr
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Translate an error code value into a text message.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00523]{DRAFT} Definition of API function `ara::diag::DiagErrorDomain::ThrowAsException` [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagErrorDomain</code></a>	
<b>Symbol:</b>	ThrowAsException(const <code>ara::core::ErrorCode</code> &errorCode)	
<b>Syntax:</b>	void ThrowAsException (const <code>ara::core::ErrorCode</code> &errorCode) const noexcept(false) override;	
<b>Parameters (in):</b>	errorCode	the <code>ErrorCode</code> instance
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept(false)	
<b>Description:</b>	Throw the exception type corresponding to the given <code>ErrorCode</code> .	

 ] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00524]{DRAFT} Definition of API function `ara::diag::GetDiagDomain` [**

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

 ] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00525]{DRAFT} Definition of API function `ara::diag::MakeErrorCode` [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	MakeErrorCode(DiagErrc code, <code>ara::core::ErrorDomain::SupportDataType</code> data)	
<b>Syntax:</b>	constexpr <code>ara::core::ErrorCode</code> MakeErrorCode (DiagErrc code, <code>ara::core::ErrorDomain::SupportDataType</code> data) noexcept;	
<b>Parameters (in):</b>	code	an enumeration value from <code>future_errc</code>
	data	a vendor-defined supplementary value
<b>Return value:</b>	<code>ara::core::ErrorCode</code>	the new <code>ErrorCode</code> instance
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Create a new <code>ErrorCode</code> for <code>DiagErrorDomain</code> with the given support data type.	

 ] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00545]{DRAFT} Definition Offer `ara::diag` errors** [Offer `ara::diag` errors shall be defined in the error domain [`ara::diag::DiagErrorDomain`](#) in accordance with [6].] ([RS\\_AP\\_00119](#))



**[SWS\_DM\_00559]{DRAFT} Definition of API enum `ara::diag::DiagOfferErrc` [**

<b>Kind:</b>	enumeration							
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"							
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"							
<b>Scope:</b>	namespace <code>ara::diag</code>							
<b>Symbol:</b>	<code>DiagOfferErrc</code>							
<b>Underlying type:</b>	<code>ara::core::ErrorDomain::CodeType</code>							
<b>Syntax:</b>	<pre>enum class DiagOfferErrc : ara::core::ErrorDomain::CodeType {...};</pre>							
<b>Values:</b>	<table border="1"> <tr> <td><code>kAlreadyOffered= 101</code></td> <td>The service is already offered.</td> </tr> <tr> <td><code>kConfiguration Mismatch= 102</code></td> <td>monitor configuration does not match dext</td> </tr> <tr> <td><code>kDebouncing Configuration Inconsistent= 103</code></td> <td>monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...</td> </tr> </table>	<code>kAlreadyOffered= 101</code>	The service is already offered.	<code>kConfiguration Mismatch= 102</code>	monitor configuration does not match dext	<code>kDebouncing Configuration Inconsistent= 103</code>	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...	
<code>kAlreadyOffered= 101</code>	The service is already offered.							
<code>kConfiguration Mismatch= 102</code>	monitor configuration does not match dext							
<code>kDebouncing Configuration Inconsistent= 103</code>	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...							
<b>Description:</b>	The <code>DiagOfferErrc</code> enumeration defines the error codes for the <code>DiagOfferErrorDomain</code> .							

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00125](#))

**[SWS\_DM\_01005]{DRAFT} Definition of API function `ara::diag::MakeErrorCode` [**

<b>Kind:</b>	function					
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"					
<b>Scope:</b>	namespace <code>ara::diag</code>					
<b>Symbol:</b>	<code>MakeErrorCode(DiagOfferErrc code, ara::core::ErrorDomain::SupportDataType data)</code>					
<b>Syntax:</b>	<pre>constexpr ara::core::ErrorCode MakeErrorCode (DiagOfferErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;</pre>					
<b>Parameters (in):</b>	<table border="1"> <tr> <td><code>code</code></td> <td>an enumeration value from <code>future_errc</code></td> </tr> <tr> <td><code>data</code></td> <td>a vendor-defined supplementary value</td> </tr> </table>	<code>code</code>	an enumeration value from <code>future_errc</code>	<code>data</code>	a vendor-defined supplementary value	
<code>code</code>	an enumeration value from <code>future_errc</code>					
<code>data</code>	a vendor-defined supplementary value					
<b>Return value:</b>	<code>ara::core::ErrorCode</code>	the new <code>ErrorCode</code> instance				
<b>Exception Safety:</b>	noexcept					
<b>Description:</b>	Create a new <code>ErrorCode</code> for <code>DiagOfferErrorDomain</code> with the given support data type.					

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00991]{DRAFT} Definition of API type `ara::diag::DiagOfferErrorDomain::Exception` [**

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00990]{DRAFT} Definition of API type ara::diag::DiagOfferErrorDomain::Errc** [

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_01003]{DRAFT} Definition of API function ara::diag::GetDiagOfferDomain** [

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

]()

**[SWS\_DM\_00995]{DRAFT} Definition of API function ara::diag::DiagOfferErrorDomain::ThrowAsException** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagOfferErrorDomain</a>
<b>Symbol:</b>	ThrowAsException(const ara::core::ErrorCode &errorCode)
<b>Syntax:</b>	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;
<b>Parameters (in):</b>	errorCode the ErrorCode instance
<b>Return value:</b>	None
<b>Exception Safety:</b>	noexcept(false)
<b>Description:</b>	Throw the exception type corresponding to the given ErrorCode.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00994]{DRAFT} Definition of API function ara::diag::DiagOfferErrorDomain::Message** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DiagOfferErrorDomain</a>	
<b>Symbol:</b>	Message(ara::core::ErrorDomain::CodeType errorCode)	
<b>Syntax:</b>	const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;	
<b>Parameters (in):</b>	errorCode	the error code value
<b>Return value:</b>	const char *	the text message, never nullptr
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Translate an error code value into a text message.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00993]{DRAFT} Definition of API function ara::diag::DiagOfferErrorDomain::Name** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DiagOfferErrorDomain</a>	
<b>Symbol:</b>	Name()	
<b>Syntax:</b>	const char * Name () const noexcept override;	
<b>Return value:</b>	const char *	"DiagOffer"
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00992]{DRAFT} Definition of API function ara::diag::DiagOfferErrorDomain::DiagOfferErrorDomain** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DiagOfferErrorDomain</a>	
<b>Symbol:</b>	DiagOfferErrorDomain()	
<b>Syntax:</b>	constexpr DiagOfferErrorDomain () noexcept;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Default constructor.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00986]{DRAFT} Definition of API function `ara::diag::DiagOfferException::DiagOfferException`** [

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

]()

**[SWS\_DM\_00989]{DRAFT} Definition of API class `ara::diag::DiagOfferErrorDomain`** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagOfferErrorDomain</code>
<b>Base class:</b>	<code>ara::core::ErrorDomain</code>
<b>Syntax:</b>	<code>class DiagOfferErrorDomain final : public ara::core::ErrorDomain {...};</code>
<b>Unique ID:</b>	0x8000'0000'0000'0403
<b>Description:</b>	Error domain for diagnostic offer errors.

] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00985]{DRAFT} Definition of API class `ara::diag::DiagOfferException`** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagOfferException</code>
<b>Base class:</b>	<code>ara::core::Exception</code>
<b>Syntax:</b>	<code>class DiagOfferException : public ara::core::Exception {...};</code>
<b>Description:</b>	Exception type thrown by <code>Diag</code> classes.

]()

**[SWS\_DM\_00546]{DRAFT} Definition Reporting `ara::diag` errors** [Reporting `ara::diag` errors shall be defined in the error domain [`ara::diag::DiagErrorDomain`](#) in accordance with [6].] ([RS\\_AP\\_00119](#))

**[SWS\_DM\_00560]{DRAFT} Definition of API enum ara::diag::DiagReportingErrc [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DiagReportingErrc	
<b>Underlying type:</b>	ara::core::ErrorDomain::CodeType	
<b>Syntax:</b>	<pre>enum class DiagReportingErrc : ara::core::ErrorDomain::CodeType {...};</pre>	
<b>Values:</b>	kAlreadyOffered= 101	The service is already offered.
	kConfiguration Mismatch= 102	monitor configuration does not match dext
	kDebouncing Configuration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
	kReportIgnored= 104	Enable Conditions disabled, OC not started, ...
	kInvalidArgument= 105	e.g. kPreFailed with internal debouncing
	kNotOffered= 106	Offer not called before reporting.
	kNoSuchDTC= 108	No mapped DTC exists for the requested event.
<b>Description:</b>	The DiagReportingErrc enumeration defines the error codes for the DiagReportingErrorDomain.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00125](#))

**[SWS\_DM\_01006]{DRAFT} Definition of API function ara::diag::MakeErrorCode [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	MakeErrorCode(DiagReportingErrc code, ara::core::ErrorDomain::SupportDataType data)	
<b>Syntax:</b>	<pre>constexpr ara::core::ErrorCode MakeErrorCode (DiagReportingErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;</pre>	
<b>Parameters (in):</b>	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
<b>Return value:</b>	ara::core::ErrorCode	the new ErrorCode instance
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Create a new ErrorCode for DiagReportingErrorDomain with the given support data type.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00998]{DRAFT} Definition of API type ara::diag::DiagReportingError Domain::Exception [**

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00997]{DRAFT} Definition of API type ara::diag::DiagReportingErrorDomain::Errc** [

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

] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_01004]{DRAFT} Definition of API function ara::diag::GetDiagReportingDomain** [

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

]()

**[SWS\_DM\_01002]{DRAFT} Definition of API function ara::diag::DiagReportingErrorDomain::ThrowAsException** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagReportingErrorDomain</a>
<b>Symbol:</b>	ThrowAsException(const ara::core::ErrorCode &errorCode)
<b>Syntax:</b>	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;
<b>Parameters (in):</b>	errorCode the ErrorCode instance
<b>Return value:</b>	None
<b>Exception Safety:</b>	noexcept(false)
<b>Description:</b>	Throw the exception type corresponding to the given ErrorCode.

] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_01001]{DRAFT} Definition of API function `ara::diag::DiagReportingErrorDomain::Message` [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagReportingErrorDomain</code></a>	
<b>Symbol:</b>	<code>Message(ara::core::ErrorDomain::CodeType errorCode)</code>	
<b>Syntax:</b>	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>	
<b>Parameters (in):</b>	<code>errorCode</code>	the error code value
<b>Return value:</b>	<code>const char *</code>	the text message, never nullptr
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Translate an error code value into a text message.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_01000]{DRAFT} Definition of API function `ara::diag::DiagReportingErrorDomain::Name` [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagReportingErrorDomain</code></a>	
<b>Symbol:</b>	<code>Name()</code>	
<b>Syntax:</b>	<code>const char * Name () const noexcept override;</code>	
<b>Return value:</b>	<code>const char *</code>	"DiagReporting"
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00999]{DRAFT} Definition of API function `ara::diag::DiagReportingErrorDomain::DiagReportingErrorDomain` [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagReportingErrorDomain</code></a>	
<b>Symbol:</b>	<code>DiagReportingErrorDomain()</code>	
<b>Syntax:</b>	<code>constexpr DiagReportingErrorDomain () noexcept;</code>	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Default constructor.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00988]{DRAFT} Definition of API function `ara::diag::DiagReportingException::DiagReportingException` [**

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

]()

**[SWS\_DM\_00996]{DRAFT} Definition of API class `ara::diag::DiagReportingErrorDomain` [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagReportingErrorDomain</code>
<b>Base class:</b>	<code>ara::core::ErrorDomain</code>
<b>Syntax:</b>	<code>class DiagReportingErrorDomain final : public ara::core::ErrorDomain {...};</code>
<b>Unique ID:</b>	0x8000'0000'0000'0402
<b>Description:</b>	Error domain for diagnostic reporting errors.

] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00987]{DRAFT} Definition of API class `ara::diag::DiagReportingException` [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagReportingException</code>
<b>Base class:</b>	<code>ara::core::Exception</code>
<b>Syntax:</b>	<code>class DiagReportingException : public ara::core::Exception {...};</code>
<b>Description:</b>	Exception type thrown by <code>Diag</code> classes.

]()

**[SWS\_DM\_00547]{DRAFT} Definition UDS NRC `ara::diag` errors [UDS NRC `ara::diag` errors shall be defined in the error domain [`ara::diag::DiagUdsNrcErrorDomain`](#) in accordance with [6].] ([RS\\_AP\\_00119](#))**



[SWS\_DM\_00526]{DRAFT} Definition of API enum `ara::diag::DiagUdsNrcErrc` [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>DiagUdsNrcErrc</code>	
<b>Underlying type:</b>	<code>int32_t</code>	
<b>Syntax:</b>	<code>enum class DiagUdsNrcErrc : int32_t {...};</code>	
<b>Values:</b>	<code>kGeneralReject= 0x10</code>	According to ISO.
	<code>kServiceNotSupported= 0x11</code>	According to ISO.
	<code>kSubfunctionNot Supported= 0x12</code>	According to ISO.
	<code>kIncorrectMessage LengthOrInvalidFormat= 0x13</code>	According to ISO.
	<code>kResponseTooLong= 0x14</code>	According to ISO.
	<code>kBusyRepeatRequest= 0x21</code>	According to ISO.
	<code>kConditionsNotCorrect= 0x22</code>	According to ISO.
	<code>kRequestSequence Error= 0x24</code>	According to ISO.
	<code>kNoResponseFrom SubnetComponent= 0x25</code>	According to ISO.
	<code>kFailurePrevents ExecutionOfRequested Action= 0x26</code>	According to ISO.
	<code>kRequestOutOfRange= 0x31</code>	According to ISO.
	<code>kSecurityAccessDenied= 0x33</code>	According to ISO.
	<code>kAuthentication Required= 0x34</code>	According to ISO.
	<code>kInvalidKey= 0x35</code>	According to ISO.
	<code>kExceedNumberOf Attempts= 0x36</code>	According to ISO.
	<code>kRequiredTimeDelayNot Expired= 0x37</code>	According to ISO.
	<code>kCertificateVerification FailedInvalidTime Period= 0x50</code>	According to ISO.
<code>kCertificateVerification FailedInvalidSignature= 0x51</code>	According to ISO.	
<code>kCertificateVerification FailedInvalidChainOf Trust= 0x52</code>	According to ISO.	
<code>kCertificateVerification FailedInvalidType= 0x53</code>	According to ISO.	





kCertificateVerificationFailedInvalidFormat=0x54	According to ISO.
kCertificateVerificationFailedInvalidContent=0x55	According to ISO.
kCertificateVerificationFailedInvalidScope=0x56	According to ISO.
kCertificateVerificationFailedInvalidCertificateRevoked=0x57	According to ISO.
kOwnershipVerificationFailed=0x58	According to ISO.
kChallengeCalculationFailed=0x59	According to ISO.
kSettingAccessRightsFailed=0x5A	According to ISO.
kSessionKeyCreationDerivationFailed=0x5B	According to ISO.
kConfigurationDataUsageFailed=0x5C	According to ISO.
kDeAuthenticationFailed=0x5D	According to ISO.
kUploadDownloadNotAccepted=0x70	According to ISO.
kTransferDataSuspended=0x71	According to ISO.
kGeneralProgrammingFailure=0x72	According to ISO.
kWrongBlockSequenceCounter=0x73	According to ISO.
kSubFunctionNotSupportedInActiveSession=0x7E	According to ISO.
kServiceNotSupportedInActiveSession=0x7F	According to ISO.
kRpmTooHigh=0x81	According to ISO.
kRpmTooLow=0x82	According to ISO.
kEnginesRunning=0x83	According to ISO.
kEnginesNotRunning=0x84	According to ISO.
kEngineRunTimeTooLow=0x85	According to ISO.
kTemperatureTooHigh=0x86	According to ISO.
kTemperatureTooLow=0x87	According to ISO.
kVehicleSpeedTooHigh=0x88	According to ISO.
kVehicleSpeedTooLow=0x89	According to ISO.
kThrottlePedalTooHigh=0x8A	According to ISO.
kThrottlePedalTooLow=0x8B	According to ISO.





	kTransmissionRangeNotInNeutral= 0x8C	According to ISO.
	kTransmissionRangeNotInGear= 0x8D	According to ISO.
	kBrakeSwitchNotClosed= 0x8F	According to ISO.
	kShifterLeverNotInPark= 0x90	According to ISO.
	kTorqueConverterClutchLocked= 0x91	According to ISO.
	kVoltageTooHigh= 0x92	According to ISO.
	kVoltageTooLow= 0x93	According to ISO.
	kResourceTemporarilyNotAvailable= 0x94	According to ISO 14229-1 Table A.1.14229-1 Table A.1.
	kNoProcessingNoResponse= 0xFF	Deviating from ISO - no further service processing and no response (silently ignore request message).
<b>Description:</b>	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.	

](RS\_AP\_00119, RS\_AP\_00125)

### [SWS\_DM\_00527]{DRAFT} Definition of API class ara::diag::DiagUdsNrcException [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DiagUdsNrcException
<b>Base class:</b>	ara::core::Exception
<b>Syntax:</b>	<code>class DiagUdsNrcException : public ara::core::Exception {...};</code>
<b>Description:</b>	Exception type thrown by Diag classes.

](RS\_AP\_00119, RS\_AP\_00132)

### [SWS\_DM\_00528]{DRAFT} Definition of API function ara::diag::DiagUdsNrcException::DiagUdsNrcException [

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

](RS\_AP\_00119, RS\_AP\_00132)

### [SWS\_DM\_00529]{DRAFT} Definition of API class `ara::diag::DiagUdsNrcErrorDomain` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/diag_uds_nrc_error_domain.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagUdsNrcErrorDomain</code>
<b>Base class:</b>	<code>ara::core::ErrorDomain</code>
<b>Syntax:</b>	<pre>class DiagUdsNrcErrorDomain final : public ara::core::ErrorDomain {...};</pre>
<b>Unique ID:</b>	0x8000'0000'0000'0411
<b>Description:</b>	Error domain for errors originating from several diagnostic classes.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00530]{DRAFT} Definition of API type `ara::diag::DiagUdsNrcErrorDomain::Errc` [

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00531]{DRAFT} Definition of API type `ara::diag::DiagUdsNrcErrorDomain::Exception` [

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00532]{DRAFT} Definition of API function `ara::diag::DiagUdsNrcErrorDomain::DiagUdsNrcErrorDomain` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/diag_uds_nrc_error_domain.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagUdsNrcErrorDomain</code></a>
<b>Symbol:</b>	<code>DiagUdsNrcErrorDomain()</code>
<b>Syntax:</b>	<code>constexpr DiagUdsNrcErrorDomain () noexcept;</code>





<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Default constructor.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00533]{DRAFT} Definition of API function `ara::diag::DiagUdsNrcErrorDomain::Name` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagUdsNrcErrorDomain</a>
<b>Symbol:</b>	Name()
<b>Syntax:</b>	<code>const char * Name () const noexcept override;</code>
<b>Return value:</b>	const char * "DiagUdsNrc"
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Return the "shortname" <code>ApApplicationErrorDomain.SN</code> of this error domain.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00534]{DRAFT} Definition of API function `ara::diag::DiagUdsNrcErrorDomain::Message` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagUdsNrcErrorDomain</a>
<b>Symbol:</b>	<code>Message(ara::core::ErrorDomain::CodeType errorCode)</code>
<b>Syntax:</b>	<code>const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;</code>
<b>Parameters (in):</b>	errorCode the error code value
<b>Return value:</b>	const char * the text message, never nullptr
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Translate an error code value into a text message.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_00535]{DRAFT} Definition of API function `ara::diag::DiagUdsNrcErrorDomain::ThrowAsException` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagUdsNrcErrorDomain</a>
<b>Symbol:</b>	<code>ThrowAsException(const ara::core::ErrorCode &amp;errorCode)</code>
<b>Syntax:</b>	<code>void ThrowAsException (const ara::core::ErrorCode &amp;errorCode) const noexcept(false) override;</code>
<b>Parameters (in):</b>	errorCode the ErrorCode instance
<b>Return value:</b>	None
<b>Exception Safety:</b>	noexcept(false)
<b>Description:</b>	Throw the exception type corresponding to the given ErrorCode.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00536]{DRAFT} Definition of API function ara::diag::GetDiagUdsNrcDomain [**

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

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_00537]{DRAFT} Definition of API function ara::diag::MakeErrorCode [**

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_uds_nrc_error_domain.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	MakeErrorCode(DiagUdsNrcErrc code, ara::core::ErrorDomain::SupportDataType data)	
<b>Syntax:</b>	constexpr ara::core::ErrorCode MakeErrorCode (DiagUdsNrcErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;	
<b>Parameters (in):</b>	code	an enumeration value from diag_errc
	data	a vendor-defined supplementary value
<b>Return value:</b>	ara::core::ErrorCode	the new ErrorCode instance
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Create a new ErrorCode for DiagUdsNrcErrorDomain with the given support data type and message.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

**[SWS\_DM\_01805]{DRAFT} Definition soVD ara::diag errors** [soVD ara::diag errors shall be defined in the error domain ara::diag::DiagSovdErrorDomain in accordance with [6].] ([RS\\_AP\\_00119](#))

**[SWS\_DM\_01806]{DRAFT} Definition of API enum ara::diag::DiagSovdErrc [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DiagSovdErrc	
<b>Underlying type:</b>	ara::core::ErrorDomain::CodeType	
<b>Syntax:</b>	enum class DiagSovdErrc : ara::core::ErrorDomain::CodeType {...};	
<b>Values:</b>	kInvalidSignature= 1	The signature of the data in the payload is invalid.
	kSovdServer Misconfigured= 2	The SOVD server is not configured correctly, e.g. required configuration files or other data is missing.





	kTemporarilyNot Available= 3	The request currently can't be served.
	kInvalidAccessToken= 4	The access token provided by the client is invalid.
	kInsufficientAccess Rights= 5	The SOVD client does not have the right to access the resource.
	kPreconditionNot Fulfilled= 6	The preconditions to execute the method are not fulfilled.
	kUpdateProcessIn Progress= 7	An update is already in progress and not yet done or aborted.
	kUpdateAutomatedNot Supported= 8	Automatic installation of update is not supported.
	kUpdatePreparationIn Progress= 9	An update is already in preparation and not yet done or aborted.
	kUpdateExecutionIn Progress= 10	Another update is currently executed and not yet done or aborted.
<b>Description:</b>	Specifies the types of errors that can occur in SOVD services.	

|(RS\_AP\_00119, RS\_AP\_00125)

### [SWS\_DM\_01807]{DRAFT} Definition of API class ara::diag::DiagSovdException

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DiagSovdException
<b>Base class:</b>	ara::core::Exception
<b>Syntax:</b>	class DiagSovdException : public ara::core::Exception {...};
<b>Description:</b>	Exception type for SOVD thrown by Diag classes.

|(RS\_AP\_00119, RS\_AP\_00132)

### [SWS\_DM\_01808]{DRAFT} Definition of API function ara::diag::DiagSovdException::DiagSovdException

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Scope:</b>	class ara::diag::DiagSovdException
<b>Symbol:</b>	DiagSovdException(ara::core::ErrorCode err)
<b>Syntax:</b>	explicit DiagSovdException (ara::core::ErrorCode err) noexcept;
<b>Parameters (in):</b>	err   the ErrorCode
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Construct a new DiagSovdException from an ErrorCode.

|(RS\_AP\_00119, RS\_AP\_00132)

### [SWS\_DM\_01809]{DRAFT} Definition of API class `ara::diag::DiagSovdErrorDomain` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DiagSovdErrorDomain</code>
<b>Base class:</b>	<code>ara::core::ErrorDomain</code>
<b>Syntax:</b>	<code>class DiagSovdErrorDomain : public ara::core::ErrorDomain {...};</code>
<b>Unique ID:</b>	0x8000'0000'0000'0412
<b>Description:</b>	Error domain for errors originating from SOVD.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_01810]{DRAFT} Definition of API type `ara::diag::DiagSovdErrorDomain::Errc` [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagSovdErrorDomain</code></a>
<b>Symbol:</b>	<code>Errc</code>
<b>Syntax:</b>	<code>using Errc = DiagSovdErrc;</code>
<b>Description:</b>	Alias for the error code value enumeration.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_01811]{DRAFT} Definition of API type `ara::diag::DiagSovdErrorDomain::Exception` [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagSovdErrorDomain</code></a>
<b>Symbol:</b>	<code>Exception</code>
<b>Syntax:</b>	<code>using Exception = DiagSovdException;</code>
<b>Description:</b>	Alias for the exception base class.

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

### [SWS\_DM\_01812]{DRAFT} Definition of API function `ara::diag::DiagSovdErrorDomain::DiagSovdErrorDomain` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::DiagSovdErrorDomain</code></a>
<b>Symbol:</b>	<code>DiagSovdErrorDomain()</code>
<b>Syntax:</b>	<code>constexpr DiagSovdErrorDomain () noexcept;</code>
<b>Exception Safety:</b>	<code>noexcept</code>
<b>Description:</b>	Default constructor.



|(RS\_AP\_00119, RS\_AP\_00132)

**[SWS\_DM\_01813]{DRAFT} Definition of API function ara::diag::DiagSovdError Domain::Name** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Scope:</b>	class ara::diag::DiagSovdErrorDomain	
<b>Symbol:</b>	Name()	
<b>Syntax:</b>	const char * Name () const noexcept override;	
<b>Return value:</b>	const char *	"DiagSovdErrorDomain"
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.	

|(RS\_AP\_00119, RS\_AP\_00132)

**[SWS\_DM\_01814]{DRAFT} Definition of API function ara::diag::DiagSovdError Domain::Message** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Scope:</b>	class ara::diag::DiagSovdErrorDomain	
<b>Symbol:</b>	Message(ara::core::ErrorDomain::CodeType errorCode)	
<b>Syntax:</b>	const char * Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;	
<b>Parameters (in):</b>	errorCode	the error code value
<b>Return value:</b>	const char *	the text message, never nullptr
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Translate an error code value into a text message.	

|(RS\_AP\_00119, RS\_AP\_00132)

**[SWS\_DM\_01815]{DRAFT} Definition of API function ara::diag::DiagSovdError Domain::ThrowAsException** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Scope:</b>	class ara::diag::DiagSovdErrorDomain	
<b>Symbol:</b>	ThrowAsException(const ara::core::ErrorCode &errorCode)	
<b>Syntax:</b>	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;	
<b>Parameters (in):</b>	errorCode	the ErrorCode instance
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept(false)	
<b>Description:</b>	Throw the exception type corresponding to the given ErrorCode.	

|(RS\_AP\_00119, RS\_AP\_00132)

## [SWS\_DM\_01816]{DRAFT} Definition of API function `ara::diag::GetDiagSovdErrorDomain` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>GetDiagSovdErrorDomain()</code>	
<b>Syntax:</b>	<code>constexpr const ara::core::ErrorDomain &amp; GetDiagSovdErrorDomain () noexcept;</code>	
<b>Return value:</b>	<code>const ara::core::ErrorDomain &amp;</code>	reference to the <code>DiagSovdErrorDomain</code> instance
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Obtain the reference to the single global <code>DiagSovdErrorDomain</code> instance.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

## [SWS\_DM\_01817]{DRAFT} Definition of API function `ara::diag::MakeErrorCode` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diag_sovd_error_domain.h"	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>MakeErrorCode(DiagSovdErrc code, ara::core::ErrorDomain::SupportDataType data)</code>	
<b>Syntax:</b>	<code>constexpr ara::core::ErrorCode MakeErrorCode (DiagSovdErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;</code>	
<b>Parameters (in):</b>	<code>code</code>	an enumeration value from <code>DiagSovdErrc</code>
	<code>data</code>	a vendor-defined supplementary value
<b>Return value:</b>	<code>ara::core::ErrorCode</code>	the new <code>ErrorCode</code> instance
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Create a new <code>ErrorCode</code> for <code>DiagSovdErrorDomain</code> with the given support data type and message.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00132](#))

## 8.3 API Reference

### 8.3.1 C++ Diagnostic API Interfaces

This chapter lists all provided and required C++ API interfaces of the `DM` for interaction with application.

#### 8.3.1.1 Introduction

Specialized `PortInterfaces` (`DiagnosticPortInterfaces`) allow an optimized deployment in the integration. In comparison of a regular `ServiceInterface` where each interface instance could be deployed individually, the diagnostic `PortInterfaces` can only be deployed for a complete process.

[SWS\_DM\_00561]{DRAFT} **Deployment of diagnostic `PortInterfaces`** [`DiagnosticPortInterfaces` shall by default interact with the machine local DM.] (*RS\_AP\_00114, RS\_Diag\_04097*)

Note: It is recommend to use `ara::com` as communication binding between the `ara::diag` library and the DM process.

Note: Platform vendors should optionally support a diagnostic deployment over machine boundaries per process (i.e. the used communication binding should support communication across machines).

The `AA` could instantiated specialized `DiagnosticPortInterfaces` for different purposes:

- `DiagnosticRoutineInterface`  
A typed interface for a single `RoutineIdentifier`
- `DiagnosticRoutineGenericInterface`  
A generic routine interface for multiple `RoutineIdentifier(s)`
- `DiagnosticDataIdentifierInterface`  
A typed data identifier interface for a single `DataIdentifier`
- `DiagnosticDataIdentifierGenericInterface`  
A generic data identifier interface for multiple `DataIdentifier(s)`
- `DiagnosticDataElementInterface`  
A typed data element interface for a single `DataElement`
- `DiagnosticGenericUdsInterface`  
A generic interface for diagnostic services
- `DiagnosticMonitorInterface`  
A interface for a single `DiagnosticEvent`

### 8.3.1.2 `MetaInfo` class

The `ara::diag::MetaInfo` class specifies a mechanism to provide meta informations, i.e. from transport protocol layer, to an interested application. To support this, `ara::diag::MetaInfo::GetValue` is specified, which provides the according value represented as a `ara::core::StringView`. The context of the current request may also be retrieved by an application by calling `ara::diag::MetaInfo::GetContext`. The context of a request could be either `kDiagnosticCommunication`, `kFaultMemory` or `kDoIP`. The detailed information on meanings of the context and key-value pairs can be looked up in [Table 8.1 “MetaInfo type definition”](#).

### 8.3.1.2.1 diag::MetalInfo class

[SWS\_DM\_00971]{DRAFT} Definition of API class `ara::diag::MetalInfo` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>MetalInfo</code>
<b>Syntax:</b>	<code>class MetalInfo final {...};</code>
<b>Description:</b>	MetaInfo interface.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.2 diag::MetalInfo::MetalInfo constructor

[SWS\_DM\_00972]{DRAFT} Definition of API function `ara::diag::MetalInfo::MetaInfo` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::MetalInfo</code></a>
<b>Symbol:</b>	<code>MetalInfo()</code>
<b>Syntax:</b>	<code>MetalInfo () noexcept=delete;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Constructor of <code>MetalInfo</code> cannot be used.

]([RS\\_Diag\\_04170](#), [RS\\_AP\\_00146](#))

### 8.3.1.2.3 diag::MetalInfo::MetalInfo copy constructor

[SWS\_DM\_00973]{DRAFT} Definition of API function `ara::diag::MetalInfo::MetaInfo` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::MetalInfo</code></a>
<b>Symbol:</b>	<code>MetalInfo(const MetalInfo &amp;)</code>
<b>Syntax:</b>	<code>MetalInfo (const MetalInfo &amp;)=delete;</code>
<b>Description:</b>	Copy Constructor of <code>MetalInfo</code> cannot be used.

]([RS\\_Diag\\_04170](#), [RS\\_AP\\_00145](#))

### 8.3.1.2.4 diag::MetalInfo::MetalInfo move constructor

[SWS\_DM\_00974]{DRAFT} Definition of API function ara::diag::MetalInfo::MetalInfo [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	class ara::diag::MetalInfo	
<b>Symbol:</b>	MetalInfo(MetalInfo &&obj)	
<b>Syntax:</b>	MetalInfo (MetalInfo &&obj) noexcept;	
<b>Parameters (in):</b>	obj	object to be moved
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move Constructor of MetalInfo.	

]([RS\\_Diag\\_04170](#), [RS\\_AP\\_00145](#))

### 8.3.1.2.5 diag::MetalInfo::MetalInfo copy assignment operator

[SWS\_DM\_00975]{DRAFT} Definition of API function ara::diag::MetalInfo::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	class ara::diag::MetalInfo	
<b>Symbol:</b>	operator=(const MetalInfo &)	
<b>Syntax:</b>	MetalInfo & operator= (const MetalInfo &)=delete;	
<b>Description:</b>	Copy Assignment Operator of MetalInfo cannot be used.	

]([RS\\_Diag\\_04170](#), [RS\\_AP\\_00145](#))

### 8.3.1.2.6 diag::MetalInfo::MetalInfo move assignment operator

[SWS\_DM\_00976]{DRAFT} Definition of API function ara::diag::MetalInfo::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	class ara::diag::MetalInfo	
<b>Symbol:</b>	operator=(MetalInfo &&other)	
<b>Syntax:</b>	MetalInfo & operator= (MetalInfo &&other) & noexcept;	
<b>Parameters (in):</b>	other	MetalInfo instance
<b>Return value:</b>	MetalInfo &	Reference to the current object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move Assignment Operator of MetalInfo.	

]([RS\\_Diag\\_04170](#), [RS\\_AP\\_00145](#))

### 8.3.1.2.7 diag::MetalInfo::Context type

[SWS\_DM\_00977]{DRAFT} Definition of API type ara::diag::MetalInfo::Context [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>
<b>Symbol:</b>	Context
<b>Syntax:</b>	using Context = std::uint32_t;
<b>Description:</b>	Definition of possible call context.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.8 diag::MetalInfo::Context kDiagnosticCommunication

[SWS\_DM\_01342]{DRAFT} Definition of API variable ara::diag::MetalInfo::kDiagnosticCommunication [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>
<b>Symbol:</b>	kDiagnosticCommunication
<b>Type:</b>	MetalInfo::Context
<b>Syntax:</b>	MetalInfo::Context kDiagnosticCommunication = 0x01;
<b>Description:</b>	Service request in DCM context.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.9 diag::MetalInfo::Context kFaultMemory

[SWS\_DM\_01343]{DRAFT} Definition of API variable ara::diag::MetalInfo::kFaultMemory [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>
<b>Symbol:</b>	kFaultMemory
<b>Type:</b>	MetalInfo::Context
<b>Syntax:</b>	MetalInfo::Context kFaultMemory = 0x02;
<b>Description:</b>	For DIDs in Snapshots.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.10 diag::MetalInfo::Context kDoIP

[SWS\_DM\_01344]{DRAFT} Definition of API variable ara::diag::MetalInfo::kDoIP [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>
<b>Symbol:</b>	kDoIP
<b>Type:</b>	MetalInfo::Context
<b>Syntax:</b>	MetaInfo::Context kDoIP = 0x03;
<b>Description:</b>	For reading VIN.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.11 diag::MetalInfo::Context kSovd

[SWS\_DM\_01818]{DRAFT} Definition of API variable ara::diag::MetalInfo::kSovd [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/meta_info.h"
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>
<b>Symbol:</b>	kSovd
<b>Type:</b>	MetalInfo::Context
<b>Syntax:</b>	MetaInfo::Context kSovd = 0x04;
<b>Description:</b>	Service request in SOVD context.

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.12 diag::MetalInfo::GetValue function

[SWS\_DM\_00978]{DRAFT} Definition of API function ara::diag::MetalInfo::GetValue [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	<a href="#">class ara::diag::MetalInfo</a>	
<b>Symbol:</b>	GetValue(ara::core::StringView key)	
<b>Syntax:</b>	ara::core::Optional< ara::core::StringView > GetValue (ara::core::StringView key) const noexcept;	
<b>Parameters (in):</b>	key	identification of value to be returned
<b>Return value:</b>	ara::core::Optional< ara::core::StringView >	Returns value for the given key.
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Get the metainfo value for a given key.	

]([RS\\_Diag\\_04170](#))

### 8.3.1.2.13 diag::MetalInfo::GetContext function

[SWS\_DM\_00979]{DRAFT} Definition of API function ara::diag::MetalInfo::GetContext [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	class ara::diag::MetalInfo	
<b>Symbol:</b>	GetContext()	
<b>Syntax:</b>	Context GetContext () const noexcept;	
<b>Return value:</b>	Context	Returns the context.
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Get the context of the invocation.	

](RS\_Diag\_04170)

### 8.3.1.2.14 diag::MetalInfo::~~MetalInfo destructor

[SWS\_DM\_00980]{DRAFT} Definition of API function ara::diag::MetalInfo::~~MetalInfo [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/meta_info.h"	
<b>Scope:</b>	class ara::diag::MetalInfo	
<b>Symbol:</b>	~MetalInfo()	
<b>Syntax:</b>	~MetalInfo () noexcept;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Default destructor.	

](RS\_Diag\_04170, RS\_AP\_00145)

Key	Context		String-Format	Description
kSA	kDiagnostic-Communication		[0-9A-F]{4}	UDS Source Address from which the diagnostic request has been sent. The value is formatted as hexadecimal number. For example tester SA of decimal 240 will have the stringified value "00F0".
kTA	kDiagnostic-Communication		[0-9A-F]{4}	UDS Target Address to which the diagnostic request has been sent. The value is formatted as hexadecimal number. For example TA of decimal 59 will have the stringified value "003B".
kTAType	kDiagnostic-Communication		"PHYS" or "FUNC"	Indicator whether request is functional or physical addressed.
kSuppPosResponse	kDiagnostic-Communication		"TRUE" or "FALSE"	Indicator whether the current request message contains a request to suppress the positive response message.
kSecurity-TimerElapsed	kDiagnostic-Communication		"TRUE" or "FALSE"	Indicates to an application, if the Security Delay Timer is elapsed.







kRe-questHandle	kDiagnostic-Communication, kSovd		[0-9]{1,5}	Key for the RequestHandle, which shall be identical for all API calls within the context of the same diagnostic request. E.g. for a Validate() and an final Confirmation() call in the context of the same diagnostic request, the same value for this key has to be placed in the metaInfo.
kLocalIP	kDiagnostic-Communication		IPv4 or IPv6 address	Key for the local IP address on which the current request gets received (this might be of interest in case the ECU is multi-homed and could receive diagnostic requests via DoIP on different interfaces). The value will be either a string in IPv4 address notation (decimal representation of address parts separated with ".") or a string in IPv6 notation (hexadecimal representation of address parts separated with ":" according to section 2.2 of RFC 4291
kLocalPort	kDiagnostic-Communication		[0-9]{1,5}	Key for the local port number on which the current request gets received. The value will be the stringified decimal representation of the port number.
kRemoteIP	kDiagnostic-Communication		IPv4 or IPv6 address	Key for the remote IP address on which the current request gets received. The value will be either a string in IPv4 address notation (decimal representation of address parts separated with ".") or a string in IPv6 notation (hexadecimal representation of address parts separated with ":" according to section 2.2 of RFC 4291)
kRemotePort	kDiagnostic-Communication		[0-9]{1,5}	Key for the remote port number on which the current request gets received. The value will be the stringified decimal representation of the port number.
kDtc		kFaultMemory	[0-9A-F]{6}	DTC number for which this interface is triggered
kBaseUri		kSOVD	URI	Base URI used for the request.
kEntityPath		kSOVD		Entity path used for the request.
kResourcePath		kSOVD		Path to the requested resource or resource collection.
kClientIdentity		kSOVD		Identity of the client. String format is defined by application handling the authentication.

**Table 8.1: MetaInfo type definition**

### 8.3.1.3 ReentrancyType class

This chapter describes the feature reentrance, which gives the [AA](#) the option to choose, whether it is called reentrant or not. Additionally this chapter defines the behaviour of the [DM](#) connected to an [AA](#) via a service, which is reentrant or not reentrant implemented. Essential to the [DM](#) is, that the [DM](#) knows about the reentrance functionality of each service and it knows, if there is an ongoing request regarding a service. With this knowledge the [DM](#) can already refuse a request of a second client to a service, which is not reentrant implemented.

**[SWS\_DM\_00935]{DRAFT} Definition of API enum ara::diag::ReentrancyType [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/reentrancy.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	ReentrancyType	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class ReentrancyType : std::uint8_t {...};	
<b>Values:</b>	kFully= 0x00	The code is fully reentrant.
	kNot= 0x01	Not reentrant code.
<b>Description:</b>	Specifies the reentrancy types.	

]([RS\\_Diag\\_04166](#))

**8.3.1.3.1 diag::DataIdentifierReentrancyType class**

**[SWS\_DM\_00936]{DRAFT} Definition of API class ara::diag::DataIdentifierReentrancyType [**

<b>Kind:</b>	struct	
<b>Header file:</b>	#include "ara/diag/reentrancy.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DataIdentifierReentrancyType	
<b>Syntax:</b>	struct DataIdentifierReentrancyType {...};	
<b>Description:</b>	Specifies the reentrancy type of a DataIdentifier related port.	

]([RS\\_Diag\\_04166](#))

**[SWS\_DM\_00937]{DRAFT} Definition of API variable ara::diag::DataIdentifierReentrancyType::read [**

<b>Kind:</b>	variable	
<b>Header file:</b>	#include "ara/diag/reentrancy.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::DataIdentifierReentrancyType</a>	
<b>Symbol:</b>	read	
<b>Type:</b>	<a href="#">ReentrancyType</a>	
<b>Syntax:</b>	ReentrancyType read;	
<b>Description:</b>	Reentrancy type for Reads.	

]([RS\\_Diag\\_04166](#))

**[SWS\_DM\_00938]{DRAFT} Definition of API variable `ara::diag::DataIdentifierReentrancyType::write` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/reentrancy.h"
<b>Scope:</b>	<a href="#">struct ara::diag::DataIdentifierReentrancyType</a>
<b>Symbol:</b>	write
<b>Type:</b>	<a href="#">ReentrancyType</a>
<b>Syntax:</b>	ReentrancyType write;
<b>Description:</b>	Reentrancy type for Writes.

]([RS\\_Diag\\_04166](#))

**[SWS\_DM\_00939]{DRAFT} Definition of API variable `ara::diag::DataIdentifierReentrancyType::readWrite` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/reentrancy.h"
<b>Scope:</b>	<a href="#">struct ara::diag::DataIdentifierReentrancyType</a>
<b>Symbol:</b>	readWrite
<b>Type:</b>	<a href="#">ReentrancyType</a>
<b>Syntax:</b>	ReentrancyType readWrite;
<b>Description:</b>	Reentrancy type for Reads and Writes.

]([RS\\_Diag\\_04166](#))

### 8.3.1.4 Monitor class

The constructor offers the possibility to add the debouncing options [ara::diag::CounterBased](#) or [ara::diag::TimeBased](#). The specifiers of the monitor constructors are only compatible with [PortInterface](#) of [DiagnosticMonitorInterface](#).

**[SWS\_DM\_00542]{DRAFT} Definition of API class `ara::diag::Monitor` [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	Monitor
<b>Syntax:</b>	class Monitor final {...};
<b>Description:</b>	Class to implement operations on diagnostic Monitor interface.

]([RS\\_Diag\\_04179](#))

### 8.3.1.4.1 diag::Monitor::Monitor constructors

#### [SWS\_DM\_00548]{DRAFT} Definition of API function ara::diag::Monitor::Monitor

[

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/monitor.h"	
<b>Scope:</b>	class ara::diag::Monitor	
<b>Symbol:</b>	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter)	
<b>Syntax:</b>	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetectionCounter	Possibility to register a function to get the current FDC for this event.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Monitor constructor for Monitors with Monitor-internal debouncing.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04179](#), [RS\\_AP\\_00121](#))

#### [SWS\_DM\_00549]{DRAFT} Definition of API function ara::diag::Monitor::Monitor

[

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/monitor.h"	
<b>Scope:</b>	class ara::diag::Monitor	
<b>Symbol:</b>	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues)	
<b>Syntax:</b>	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Monitor constructor for Monitors with counter-based debouncing.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04179](#), [RS\\_Diag\\_04068](#), [RS\\_AP\\_00121](#))

#### [SWS\_DM\_00550]{DRAFT} Definition of API function ara::diag::Monitor::Monitor

[

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/monitor.h"	
<b>Scope:</b>	class ara::diag::Monitor	
<b>Symbol:</b>	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, TimeBased defaultValues)	





<b>Syntax:</b>	Monitor (const ara::core::InstanceSpecifier &specifier, std::function<void(InitMonitorReason)> initMonitor, TimeBased defaultValues);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitor Interface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Monitor constructor for Monitors with time-based debouncing.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04179](#), [RS\\_Diag\\_04225](#), [RS\\_AP\\_00121](#))

### 8.3.1.4.2 diag::Monitor::ReportMonitorAction function

[SWS\_DM\_00543]{DRAFT} Definition of API function ara::diag::Monitor::ReportMonitorAction [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/monitor.h"	
<b>Scope:</b>	class ara::diag::Monitor	
<b>Symbol:</b>	ReportMonitorAction(MonitorAction action)	
<b>Syntax:</b>	void ReportMonitorAction (MonitorAction action);	
<b>Parameters (in):</b>	action	Contains either the last (un-)qualified test result of the diagnostic monitor or commands to control the debouncing or to force a prestorage.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Function to report the status information being relevant for error monitoring paths.	

]([RS\\_Diag\\_04179](#), [RS\\_AP\\_00139](#), [RS\\_AP\\_00119](#))

### 8.3.1.4.3 diag::Monitor::Offer function

[SWS\_DM\_01088]{DRAFT} Definition of API function ara::diag::Monitor::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/monitor.h"	
<b>Scope:</b>	class ara::diag::Monitor	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	With the Offer the application states that it is ready to receive and process initMonitor callouts.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.1.4.4 diag::Monitor::StopOffer function

[SWS\_DM\_01089]{DRAFT} Definition of API function ara::diag::Monitor::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::Monitor</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04169](#))

### 8.3.1.4.5 diag::Monitor::operator= move assignment operator

[SWS\_DM\_01684]{DRAFT} Definition of API function ara::diag::Monitor::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::Monitor</a>
<b>Symbol:</b>	operator=(Monitor &&)
<b>Syntax:</b>	<a href="#">Monitor</a> & operator= ( <a href="#">Monitor</a> &&)=delete;
<b>Description:</b>	Move assignment operator of Monitor.

]([RS\\_AP\\_00147](#))

### 8.3.1.4.6 diag::Monitor::operator= copy assignment operator

[SWS\_DM\_01683]{DRAFT} Definition of API function ara::diag::Monitor::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::Monitor</a>
<b>Symbol:</b>	operator=(Monitor &)
<b>Syntax:</b>	<a href="#">Monitor</a> & operator= ( <a href="#">Monitor</a> &)=delete;
<b>Description:</b>	Monitor shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.4.7 diag::Monitor::Monitor(Monitor&&) move constructor

[SWS\_DM\_01686]{DRAFT} Definition of API function ara::diag::Monitor::Monitor

[

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::Monitor</a>
<b>Symbol:</b>	Monitor(Monitor &&)
<b>Syntax:</b>	Monitor (Monitor &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of Monitor.

]([RS\\_AP\\_00147](#))

### 8.3.1.4.8 diag::Monitor::Monitor(Monitor&) copy constructor

[SWS\_DM\_01685]{DRAFT} Definition of API function ara::diag::Monitor::Monitor

[

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::Monitor</a>
<b>Symbol:</b>	Monitor(Monitor &)
<b>Syntax:</b>	Monitor (Monitor &)=delete;
<b>Description:</b>	Monitor shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.5 Monitor types class

The constructor offers the possibility to add the debouncing options [ara::diag::CounterBased](#) or [ara::diag::TimeBased](#). The specifiers of the monitor constructors are only compatible with [PortInterface](#) of [DiagnosticMonitorInterface](#).

#### 8.3.1.5.1 diag::CounterBased type

[SWS\_DM\_00538]{DRAFT} Definition of API class ara::diag::CounterBased [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"





<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	CounterBased
<b>Syntax:</b>	struct CounterBased {...};
<b>Description:</b>	Represents the parameters for counter-based debouncing.

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00621]{DRAFT} Definition of API variable ara::diag::CounterBased::failedThreshold [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	failedThreshold
<b>Type:</b>	std::int16_t
<b>Syntax:</b>	std::int16_t failedThreshold;
<b>Description:</b>	Threshold until qualified failed.

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00622]{DRAFT} Definition of API variable ara::diag::CounterBased::passedThreshold [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	passedThreshold
<b>Type:</b>	std::int16_t
<b>Syntax:</b>	std::int16_t passedThreshold;
<b>Description:</b>	Threshold until qualified passed.

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00623]{DRAFT} Definition of API variable ara::diag::CounterBased::failedStepsize [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	failedStepsize
<b>Type:</b>	std::uint16_t
<b>Syntax:</b>	std::uint16_t failedStepsize;
<b>Description:</b>	Stepsize per pre-failed report.

]([RS\\_Diag\\_04068](#))



**[SWS\_DM\_00624]{DRAFT} Definition of API variable ara::diag::CounterBased::passedStepsize [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	passedStepsize
<b>Type:</b>	std::uint16_t
<b>Syntax:</b>	std::uint16_t passedStepsize;
<b>Description:</b>	Stepsize per pre-passed report.

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00625]{DRAFT} Definition of API variable ara::diag::CounterBased::failedJumpValue [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	failedJumpValue
<b>Type:</b>	std::int16_t
<b>Syntax:</b>	std::int16_t failedJumpValue;
<b>Description:</b>	failed to jump value

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00626]{DRAFT} Definition of API variable ara::diag::CounterBased::passedJumpValue [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	passedJumpValue
<b>Type:</b>	std::int16_t
<b>Syntax:</b>	std::int16_t passedJumpValue;
<b>Description:</b>	passed to jump value

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00627]{DRAFT} Definition of API variable ara::diag::CounterBased::useJumpToFailed [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	useJumpToFailed
<b>Type:</b>	bool
<b>Syntax:</b>	bool useJumpToFailed;
<b>Description:</b>	is jump supported

]([RS\\_Diag\\_04068](#))

**[SWS\_DM\_00628]{DRAFT} Definition of API variable ara::diag::CounterBased::useJumpToPassed** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CounterBased</a>
<b>Symbol:</b>	useJumpToPassed
<b>Type:</b>	bool
<b>Syntax:</b>	bool useJumpToPassed;
<b>Description:</b>	is jump supported

] ([RS\\_Diag\\_04068](#))

### 8.3.1.5.2 diag::TimeBased type

**[SWS\_DM\_00539]{DRAFT} Definition of API class ara::diag::TimeBased** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	TimeBased
<b>Syntax:</b>	struct TimeBased {...};
<b>Description:</b>	Represents the parameters for time-based debouncing.

] ([RS\\_Diag\\_04225](#))

**[SWS\_DM\_00630]{DRAFT} Definition of API variable ara::diag::TimeBased::passedMs** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::TimeBased</a>
<b>Symbol:</b>	passedMs
<b>Type:</b>	std::uint32_t
<b>Syntax:</b>	std::uint32_t passedMs;
<b>Description:</b>	time until passed in (ms)

] ([RS\\_Diag\\_04225](#))

**[SWS\_DM\_00629]{DRAFT} Definition of API variable ara::diag::TimeBased::failedMs** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/monitor_types.h"
<b>Scope:</b>	<a href="#">struct ara::diag::TimeBased</a>





<b>Symbol:</b>	failedMs
<b>Type:</b>	std::uint32_t
<b>Syntax:</b>	std::uint32_t failedMs;
<b>Description:</b>	time until failed in (ms)

|(RS\_Diag\_04225)

### 8.3.1.5.3 diag::InitMonitorReason enum

[SWS\_DM\_00540]{DRAFT} Definition of API enum ara::diag::InitMonitorReason [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/monitor_types.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	InitMonitorReason	
<b>Underlying type:</b>	std::uint32_t	
<b>Syntax:</b>	enum class InitMonitorReason : std::uint32_t {...};	
<b>Values:</b>	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
<b>Description:</b>	Represents the status information reported to AAs why the monitor may be re-initialized.	

|(RS\_Diag\_04179)

### 8.3.1.5.4 diag::MonitorAction enum

[SWS\_DM\_00541]{DRAFT} Definition of API enum ara::diag::MonitorAction [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/monitor_types.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	MonitorAction	
<b>Underlying type:</b>	std::uint32_t	
<b>Syntax:</b>	enum class MonitorAction : std::uint32_t {...};	
<b>Values:</b>	kPassed= 0x00	Monitor reports qualified test result passed.
	kFailed= 0x01	Monitor reports qualified test result failed.
	kPrepassed= 0x02	Monitor reports unqualified test result pre-passed.
	kPrefailed= 0x03	Monitor reports unqualified test result pre-failed.





	kFdcThreshold Reached= 0x04	Monitor triggers the storage of ExtendedDataRecords and Freeze Frames (if the triggering condition is connected to this threshold).
	kResetTestFailed= 0x05	Reset TestFailed Bit without any other side effects like readiness
	kFreezeDebouncing= 0x06	Freeze the internal debounce counter/timer.
	kResetDebouncing= 0x07	Reset the internal debounce counter/timer.
<b>Description:</b>	Represents the status information reported by AAs being relevant for error monitoring.	

|(RS\_Diag\_04179)

### 8.3.1.6 MultipleMonitor class

The conversation object can only be retrieved by a given meta\_info object.

[SWS\_DM\_01694]{DRAFT} Definition of API class ara::diag::MultipleMonitor [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	MultipleMonitor
<b>Syntax:</b>	<code>class MultipleMonitor final {...};</code>
<b>Description:</b>	Class to implement monitor operations on the multiple monitor interfaces.

|(RS\_Diag\_04179)

#### 8.3.1.6.1 diag::MonitorHandleType type

[SWS\_DM\_01701]{DRAFT} Definition of API type ara::diag::MonitorHandleType [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	MonitorHandleType
<b>Syntax:</b>	<code>using MonitorHandleType = std::uint32_t;</code>
<b>Description:</b>	Type to identify an monitor .

|(RS\_Diag\_04063)

### 8.3.1.6.2 diag::MultipleMonitor::MultipleMonitor type

[SWS\_DM\_01695]{DRAFT} Definition of API function ara::diag::MultipleMonitor::MultipleMonitor [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	MultipleMonitor(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	MultipleMonitor (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMultipleMonitorInterface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	MultipleMonitor constructor allows to report status of more than one monitor.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04179](#), [RS\\_AP\\_00121](#))

### 8.3.1.6.3 diag::MultipleMonitor::ConfigureMonitor function

[SWS\_DM\_01696]{DRAFT} Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	ConfigureMonitor(MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t(> getFaultDetectionCounter)	
<b>Syntax:</b>	ara::core::Result< void > ConfigureMonitor (MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t(> getFaultDetectionCounter);	
<b>Parameters (in):</b>	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.
<b>Return value:</b>	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided monitorHandle is invalid/unknown
<b>Description:</b>	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.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04179](#), [RS\\_AP\\_00121](#))

### 8.3.1.6.4 diag::MultipleMonitor::ConfigureMonitor function

[SWS\_DM\_01702]{DRAFT} Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	ConfigureMonitor(MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues)	
<b>Syntax:</b>	<pre>ara::core::Result&lt; void &gt; ConfigureMonitor (MonitorHandleType monitorHandle, std::function&lt; void(InitMonitorReason)&gt; initMonitor, CounterBased defaultValues);</pre>	
<b>Parameters (in):</b>	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
<b>Return value:</b>	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided monitorHandle is invalid/unknown
<b>Description:</b>	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.	

](RS\_AP\_00137, RS\_Diag\_04179, RS\_Diag\_04068, RS\_AP\_00121)

### 8.3.1.6.5 diag::MultipleMonitor::ConfigureMonitor function

[SWS\_DM\_01697]{DRAFT} Definition of API function ara::diag::MultipleMonitor::ConfigureMonitor [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	ConfigureMonitor(MonitorHandleType monitorHandle, std::function< void(InitMonitorReason)> initMonitor, TimeBased defaultValues)	
<b>Syntax:</b>	<pre>ara::core::Result&lt; void &gt; ConfigureMonitor (MonitorHandleType monitorHandle, std::function&lt; void(InitMonitorReason)&gt; initMonitor, TimeBased defaultValues);</pre>	
<b>Parameters (in):</b>	monitorHandle	Identifies the requested monitor
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
<b>Return value:</b>	ara::core::Result< void >	Failure if any overload of ConfigureMonitor was called already
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided monitorHandle is invalid/unknown
<b>Description:</b>	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.	

](RS\_AP\_00137, RS\_Diag\_04179, RS\_Diag\_04225, RS\_AP\_00121)

### 8.3.1.6.6 diag::MultipleMonitor::ReportMonitorAction function

[SWS\_DM\_01698]{DRAFT} Definition of API function ara::diag::MultipleMonitor::ReportMonitorAction [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	ReportMonitorAction(MonitorHandleType monitorHandle, MonitorAction action)	
<b>Syntax:</b>	void ReportMonitorAction (MonitorHandleType monitorHandle, MonitorAction action);	
<b>Parameters (in):</b>	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.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Function to report the status information being relevant for error monitoring paths. Calls with invalid monitorHandle are ignored.	

](RS\_Diag\_04179, RS\_AP\_00139, RS\_AP\_00119)

### 8.3.1.6.7 diag::MultipleMonitor::Offer function

[SWS\_DM\_01699]{DRAFT} Definition of API function ara::diag::MultipleMonitor::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"	
<b>Scope:</b>	class ara::diag::MultipleMonitor	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

](RS\_AP\_00119, RS\_AP\_00139, RS\_Diag\_04169)

### 8.3.1.6.8 diag::MultipleMonitor::StopOffer function

[SWS\_DM\_01700]{DRAFT} Definition of API function ara::diag::MultipleMonitor::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/multiple_monitor.h"
<b>Scope:</b>	<a href="#">class ara::diag::MultipleMonitor</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04169](#))

### 8.3.1.7 GenericUDSService class

This interface allows a generic implementation to handle UDS messages. Several [DiagnosticServiceGenericMappings](#) with a reference to [diagnosticServiceInstance](#) can map to the same [PortPrototype](#). The [specifier](#) is only compatible with [PortInterface](#) of [DiagnosticGenericUdsInterface](#).

[SWS\_DM\_00602]{DRAFT} Definition of API class ara::diag::GenericUDSService [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	GenericUDSService
<b>Syntax:</b>	class GenericUDSService {...};
<b>Description:</b>	Generic UDS interface.

]([RS\\_Diag\\_04169](#))

#### 8.3.1.7.1 diag::GenericUDSService::OperationOutput type

[SWS\_DM\_00578]{DRAFT} Definition of API class ara::diag::GenericUDSService::OperationOutput [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"







<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	OperationOutput
<b>Syntax:</b>	struct OperationOutput {...};
<b>Description:</b>	Response data of positive response message.

|(RS\_Diag\_04169, RS\_Diag\_04172)

### 8.3.1.7.2 diag::GenericUDSService::OperationOutput::responseData

[SWS\_DM\_00632]{DRAFT} Definition of API variable `ara::diag::GenericUDSService::OperationOutput::responseData` [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">struct ara::diag::GenericUDSService::OperationOutput</a>
<b>Symbol:</b>	responseData
<b>Type:</b>	ara::core::Vector<std::uint8_t>
<b>Syntax:</b>	ara::core::Vector<std::uint8_t> responseData;
<b>Description:</b>	Content of positive response message (without SID)

|(RS\_Diag\_04169, RS\_Diag\_04172)

### 8.3.1.7.3 diag::GenericUDSService::GenericUDSService function

[SWS\_DM\_00616]{DRAFT} Definition of API function `ara::diag::GenericUDSService::GenericUDSService` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"	
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>	
<b>Symbol:</b>	GenericUDSService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit GenericUDSService (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of GenericUDSService.	

|(RS\_AP\_00137, RS\_Diag\_04169)

### 8.3.1.7.4 diag::GenericUDSService::~~GenericUDSService function

[SWS\_DM\_00584]{DRAFT} Definition of API function `ara::diag::GenericUDSService::~~GenericUDSService` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	~GenericUDSService()
<b>Syntax:</b>	virtual ~GenericUDSService () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of GenericUDSService.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04169](#))

### 8.3.1.7.5 diag::GenericUDSService::GenericUDSService(GenericUDSService&&) move constructor

[SWS\_DM\_01658]{DRAFT} Definition of API function `ara::diag::GenericUDSService::GenericUDSService` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	GenericUDSService(GenericUDSService &&)
<b>Syntax:</b>	GenericUDSService (GenericUDSService &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of GenericUDSService.

]([RS\\_AP\\_00147](#))

### 8.3.1.7.6 diag::GenericUDSService::GenericUDSService(GenericUDSService&) copy constructor

[SWS\_DM\_01657]{DRAFT} Definition of API function `ara::diag::GenericUDSService::GenericUDSService` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	GenericUDSService(GenericUDSService &)
<b>Syntax:</b>	GenericUDSService (GenericUDSService &)=delete;
<b>Description:</b>	GenericUDSService shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.7.7 diag::GenericUDSService::GenericUDSService& operator=(GenericUDSService&&) move operator opera-

[SWS\_DM\_01656]{DRAFT} Definition of API function ara::diag::GenericUDSService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	operator=(GenericUDSService &&)
<b>Syntax:</b>	<a href="#">GenericUDSService</a> & operator= ( <a href="#">GenericUDSService</a> &&)=delete;
<b>Description:</b>	Move assignment operator of GenericUDSService.

]([RS\\_AP\\_00147](#))

### 8.3.1.7.8 diag::GenericUDSService::GenericUDSService& operator=(GenericUDSService&) copy operator opera-

[SWS\_DM\_01655]{DRAFT} Definition of API function ara::diag::GenericUDSService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	operator=(GenericUDSService &)
<b>Syntax:</b>	<a href="#">GenericUDSService</a> & operator= ( <a href="#">GenericUDSService</a> &)=delete;
<b>Description:</b>	GenericUDSService shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.7.9 diag::GenericUDSService::Offer function

[SWS\_DM\_00619]{DRAFT} Definition of API function ara::diag::GenericUDSService::Offer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	Offer()
<b>Syntax:</b>	ara::core::Result< void > Offer ();
<b>Return value:</b>	ara::core::Result< void >   -
<b>Exception Safety:</b>	not exception safe





<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00139](#), [RS\\_AP\\_00119](#), [RS\\_Diag\\_04169](#))

### 8.3.1.7.10 diag::GenericUDSService::StopOffer function

[SWS\_DM\_00620][DRAFT] Definition of API function ara::diag::GenericUDSService::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04169](#))

### 8.3.1.7.11 diag::GenericUDSService::HandleMessage function

[SWS\_DM\_00618][DRAFT] Definition of API function ara::diag::GenericUDSService::HandleMessage [

<b>Kind:</b>	function								
<b>Header file:</b>	#include "ara/diag/generic_uds_service.h"								
<b>Scope:</b>	<a href="#">class ara::diag::GenericUDSService</a>								
<b>Symbol:</b>	HandleMessage(std::uint8_t sid, ara::core::Span< const std::uint8_t > requestData, const Meta Info &metaInfo, CancellationHandler cancellationHandler)								
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > HandleMessage (std::uint8_t sid, ara::core::Span< const std::uint8_t > requestData, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;								
<b>Parameters (in):</b>	<table border="1"> <tr> <td>sid</td> <td>Diagnostic Request Service Identifier.</td> </tr> <tr> <td>requestData</td> <td>Diagnostic request data (starting after SID).</td> </tr> <tr> <td>metaInfo</td> <td>MetaInfo of the request.</td> </tr> <tr> <td>cancellationHandler</td> <td>This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..</td> </tr> </table>	sid	Diagnostic Request Service Identifier.	requestData	Diagnostic request data (starting after SID).	metaInfo	MetaInfo of the request.	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
sid	Diagnostic Request Service Identifier.								
requestData	Diagnostic request data (starting after SID).								
metaInfo	MetaInfo of the request.								
cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..								





<b>Return value:</b>	ara::core::Future<OperationOutput >	a Result with either a OperationOutput or an error
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResource TemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
<b>Description:</b>	Called for any request message.

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04172, RS\_Diag\_04173)

### 8.3.1.8 GenericDataIdentifier class

This interface allows a generic implementation of a data identifier handler. Multiple [DiagnosticServiceGenericMappings](#) can reference to the same [PortPrototype](#). The `specifier` is only compatible with [PortInterface](#) of [DiagnosticDataIdentifierGenericInterface](#).

**[SWS\_DM\_00607]{DRAFT} Definition of API class ara::diag::GenericDataIdentifier** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	GenericDataIdentifier
<b>Syntax:</b>	class GenericDataIdentifier {...};
<b>Description:</b>	Generic DataIdentifier interface.

]([RS\\_Diag\\_04169](#))

### 8.3.1.8.1 diag::GenericDataIdentifier::OperationOutput type

**[SWS\_DM\_00641]{DRAFT} Definition of API class ara::diag::GenericDataIdentifier::OperationOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericDataIdentifier</a>
<b>Symbol:</b>	OperationOutput
<b>Syntax:</b>	struct OperationOutput {...};
<b>Description:</b>	Response data of positive response message.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04172](#))

**[SWS\_DM\_00631]{DRAFT} Definition of API variable ara::diag::GenericDataIdentifier::OperationOutput::responseData** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Scope:</b>	<a href="#">struct ara::diag::GenericDataIdentifier::OperationOutput</a>
<b>Symbol:</b>	responseData
<b>Type:</b>	ara::core::Vector< std::uint8_t >
<b>Syntax:</b>	ara::core::Vector<std::uint8_t> responseData;
<b>Description:</b>	Content of positive response message (without DataIdentifier)

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#))



### 8.3.1.8.2 diag::GenericDataIdentifier::GenericDataIdentifier function

[SWS\_DM\_00634]{DRAFT} Definition of API function `ara::diag::GenericDataIdentifier::GenericDataIdentifier` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"	
<b>Scope:</b>	<a href="#">class ara::diag::GenericDataIdentifier</a>	
<b>Symbol:</b>	GenericDataIdentifier(const ara::core::InstanceSpecifier &specifier, DataIdentifierReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit GenericDataIdentifier (const ara::core::InstanceSpecifier &specifier, <a href="#">DataIdentifierReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDataIdentifierGenericInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant for reads, writes or both
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Class for an GenericDataIdentifier.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#))

### 8.3.1.8.3 diag::GenericDataIdentifier::~~GenericDataIdentifier function

[SWS\_DM\_00635]{DRAFT} Definition of API function `ara::diag::GenericDataIdentifier::~~GenericDataIdentifier` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"	
<b>Scope:</b>	<a href="#">class ara::diag::GenericDataIdentifier</a>	
<b>Symbol:</b>	~GenericDataIdentifier()	
<b>Syntax:</b>	virtual ~GenericDataIdentifier () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructor of class GenericDataIdentifier.	

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04169](#))

### 8.3.1.8.4 diag::GenericDataIdentifier::GenericDataIdentifier(GenericDataIdentifier&&) move constructor

[SWS\_DM\_01654]{DRAFT} Definition of API function `ara::diag::GenericDataIdentifier::GenericDataIdentifier` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"	
<b>Scope:</b>	<a href="#">class ara::diag::GenericDataIdentifier</a>	



△

<b>Symbol:</b>	GenericDataIdentifier(GenericDataIdentifier &&)
<b>Syntax:</b>	GenericDataIdentifier (GenericDataIdentifier &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of GenericDataIdentifier.

](RS\_AP\_00147)

### 8.3.1.8.5 diag::GenericDataIdentifier::GenericDataIdentifier(GenericDataIdentifier&) copy constructor

[SWS\_DM\_01653]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::GenericDataIdentifier [

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

](RS\_AP\_00147)

### 8.3.1.8.6 diag::GenericDataIdentifier::GenericDataIdentifier& operator=(GenericDataIdentifier&&) move operator

[SWS\_DM\_01652]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Scope:</b>	class ara::diag::GenericDataIdentifier
<b>Symbol:</b>	operator=(GenericDataIdentifier &&)
<b>Syntax:</b>	GenericDataIdentifier & operator= (GenericDataIdentifier &&)=delete;
<b>Description:</b>	Move assignment operator of GenericDataIdentifier.

](RS\_AP\_00147)

### 8.3.1.8.7 diag::GenericDataIdentifier::GenericDataIdentifier& operator=(GenericDataIdentifier&&) copy operator opera-

[SWS\_DM\_01651]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Scope:</b>	class ara::diag::GenericDataIdentifier
<b>Symbol:</b>	operator=(GenericDataIdentifier &)
<b>Syntax:</b>	GenericDataIdentifier & operator= (GenericDataIdentifier &)=delete;
<b>Description:</b>	GenericDataIdentifier shall be a single not assignable instance.

] ([RS\\_AP\\_00147](#))

### 8.3.1.8.8 diag::GenericDataIdentifier::Offer function

[SWS\_DM\_00638]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::Offer [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"		
<b>Scope:</b>	class ara::diag::GenericDataIdentifier		
<b>Symbol:</b>	Offer()		
<b>Syntax:</b>	ara::core::Result< void > Offer ();		
<b>Return value:</b>	ara::core::Result< void > -		
<b>Exception Safety:</b>	not exception safe		
<b>Errors:</b>	<table border="1"> <tr> <td>DiagErrorDomain::Diag OfferErrc::kAlready Offered</td> <td>This service was already offered.</td> </tr> </table>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.		
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.		

] ([RS\\_AP\\_00139](#), [RS\\_AP\\_00119](#), [RS\\_Diag\\_04169](#))

### 8.3.1.8.9 diag::GenericDataIdentifier::StopOffer function

[SWS\_DM\_00639]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"
<b>Scope:</b>	class ara::diag::GenericDataIdentifier
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None





<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

] ([RS\\_Diag\\_04169](#))

### 8.3.1.8.10 diag::GenericDataIdentifier::Read function

[SWS\_DM\_00636]{DRAFT} Definition of API function `ara::diag::GenericDataIdentifier::Read` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"	
<b>Scope:</b>	<a href="#">class ara::diag::GenericDataIdentifier</a>	
<b>Symbol:</b>	Read(std::uint16_t dataIdentifier, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > Read (std::uint16_t dataIdentifier, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	dataIdentifier	the corresponding DataIdentifier
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< <a href="#">OperationOutput</a> >	a Result with either <a href="#">OperationOutput</a> (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResource TemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
<b>Description:</b>	Called for ReadDataByIdentifier request for this DiagnosticDataIdentifier.	

|(RS\_AP\_00138, RS\_AP\_00119, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04172)

### 8.3.1.8.11 diag::GenericDataIdentifier::Write function

[SWS\_DM\_00637]{DRAFT} Definition of API function ara::diag::GenericDataIdentifier::Write

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_data_identifier.h"	
<b>Scope:</b>	class ara::diag::GenericDataIdentifier	
<b>Symbol:</b>	Write(std::uint16_t dataIdentifier, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > Write (std::uint16_t dataIdentifier, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler);	
<b>Parameters (in):</b>	dataIdentifier	the corresponding DataIdentifier
	requestData	Content of request message (without DataIdentifier)
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	





<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.







DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for WriteDataByIdentifier request for this DiagnosticDataIdentifier.	

]([RS\\_AP\\_00138](#), [RS\\_AP\\_00119](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04170](#))

### 8.3.1.9 GenericRoutine class

This interface allows a generic implementation of a routine handler. Several `DiagnosticServiceGenericMappings` with a reference to `diagnosticServiceInstance` can map to the same `PortPrototype`. The `specifier` is only compatible with `PortInterface` of `DiagnosticRoutineGenericInterface`.

**[SWS\_DM\_00605]{DRAFT} Definition of API class `ara::diag::GenericRoutine` [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>GenericRoutine</code>





<b>Syntax:</b>	<code>class GenericRoutine {...};</code>
<b>Description:</b>	Generic Routine interface.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.1 diag::GenericRoutine::OperationOutput type

[SWS\_DM\_00551]{DRAFT} **Definition of API class ara::diag::GenericRoutine::OperationOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	<code>#include "ara/diag/generic_routine.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	OperationOutput
<b>Syntax:</b>	<code>struct OperationOutput {...};</code>
<b>Description:</b>	Response data of positive response message.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

[SWS\_DM\_00633]{DRAFT} **Definition of API variable ara::diag::GenericRoutine::OperationOutput::responseData** [

<b>Kind:</b>	variable
<b>Header file:</b>	<code>#include "ara/diag/generic_routine.h"</code>
<b>Scope:</b>	<a href="#">struct ara::diag::GenericRoutine::OperationOutput</a>
<b>Symbol:</b>	responseData
<b>Type:</b>	<code>ara::core::Vector&lt;std::uint8_t&gt;</code>
<b>Syntax:</b>	<code>ara::core::Vector&lt;std::uint8_t&gt; responseData;</code>
<b>Description:</b>	Content of positive response message (without RoutineIdentifier)

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.2 diag::GenericRoutine::GenericRoutine function

[SWS\_DM\_00552]{DRAFT} **Definition of API function ara::diag::GenericRoutine::GenericRoutine** [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/generic_routine.h"</code>
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	<code>GenericRoutine(const ara::core::InstanceSpecifier &amp;specifier, ReentrancyType reentrancyType)</code>
<b>Syntax:</b>	<code>explicit GenericRoutine (const ara::core::InstanceSpecifier &amp;specifier, <a href="#">ReentrancyType</a> reentrancyType);</code>





<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticRoutine GenericInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Class for an GenericRoutine.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.3 diag::GenericRoutine::GenericRoutine(GenericRoutine&&) move constructor

[SWS\_DM\_01638]{DRAFT} Definition of API function ara::diag::GenericRoutine::GenericRoutine [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	GenericRoutine(GenericRoutine &&)
<b>Syntax:</b>	GenericRoutine (GenericRoutine &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of GenericRoutine.

]([RS\\_AP\\_00147](#))

### 8.3.1.9.4 diag::GenericRoutine::GenericRoutine(GenericRoutine&) copy constructor

[SWS\_DM\_01637]{DRAFT} Definition of API function ara::diag::GenericRoutine::GenericRoutine [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	GenericRoutine(GenericRoutine &)
<b>Syntax:</b>	GenericRoutine (GenericRoutine &)=delete;
<b>Description:</b>	GenericRoutine shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.9.5 diag::GenericRoutine::GenericRoutine& operator=(GenericRoutine&&) move operator opera-

[SWS\_DM\_01636]{DRAFT} Definition of API function ara::diag::GenericRoutine::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	operator=(GenericRoutine &&)
<b>Syntax:</b>	<a href="#">GenericRoutine</a> & operator= ( <a href="#">GenericRoutine</a> &&)=delete;
<b>Description:</b>	Move assignment operator of GenericRoutine.

]([RS\\_AP\\_00147](#))

### 8.3.1.9.6 diag::GenericRoutine::GenericRoutine& operator=(GenericRoutine&) move operator

[SWS\_DM\_01635]{DRAFT} Definition of API function ara::diag::GenericRoutine::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	operator=(GenericRoutine &)
<b>Syntax:</b>	<a href="#">GenericRoutine</a> & operator= ( <a href="#">GenericRoutine</a> &)=delete;
<b>Description:</b>	GenericRoutine shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.9.7 diag::GenericRoutine::~~GenericRoutine function

[SWS\_DM\_00553]{DRAFT} Definition of API function ara::diag::GenericRoutine::~~GenericRoutine [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/generic_routine.h"
<b>Scope:</b>	<a href="#">class ara::diag::GenericRoutine</a>
<b>Symbol:</b>	~GenericRoutine()
<b>Syntax:</b>	virtual ~GenericRoutine () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class GenericRoutine.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.8 diag::GenericRoutine::Offer function

[SWS\_DM\_00557]{DRAFT} Definition of API function `ara::diag::GenericRoutine::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_routine.h"	
<b>Scope:</b>	class <a href="#">ara::diag::GenericRoutine</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; Offer ();</code>	
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00138](#), [RS\\_Main\\_01002](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.9 diag::GenericRoutine::StopOffer function

[SWS\_DM\_00558]{DRAFT} Definition of API function `ara::diag::GenericRoutine::StopOffer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_routine.h"	
<b>Scope:</b>	class <a href="#">ara::diag::GenericRoutine</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	<code>void StopOffer ();</code>	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.1.9.10 diag::GenericRoutine::Start function

[SWS\_DM\_00554]{DRAFT} Definition of API function `ara::diag::GenericRoutine::Start` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_routine.h"	
<b>Scope:</b>	class <a href="#">ara::diag::GenericRoutine</a>	





<b>Symbol:</b>	Start(std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > Start (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier)
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< <a href="#">OperationOutput</a> >	a Result with either <a href="#">OperationOutput</a> (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesRunning	0x83, Negative return code according to ISO 14229.







DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction Start request for this DiagnosticRoutineIdentifier.

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04224)

### 8.3.1.9.11 diag::GenericRoutine::Stop function

[SWS\_DM\_00555]{DRAFT} Definition of API function ara::diag::GenericRoutine::Stop

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_routine.h"	
<b>Scope:</b>	class ara::diag::GenericRoutine	
<b>Symbol:</b>	Stop(std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > Stop (std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler);	
<b>Parameters (in):</b>	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier)
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< <a href="#">OperationOutput</a> >	a Result with either <a href="#">OperationOutput</a> (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction Stop request for this DiagnosticRoutineIdentifier.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04224)

### 8.3.1.9.12 diag::GenericRoutine::RequestResults function

[SWS\_DM\_00556][DRAFT] Definition of API function ara::diag::GenericRoutine::RequestResults [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/generic_routine.h"	
<b>Scope:</b>	class ara::diag::GenericRoutine	
<b>Symbol:</b>	RequestResults(std::uint16_t routineId, ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; <a href="#">OperationOutput</a> &gt; RequestResults (std::uint16_t routineId, ara::core::Span&lt; const std::uint8_t &gt; requestData, const <a href="#">MetaInfo</a> &amp;metaInfo, <a href="#">CancellationHandler</a> cancellationHandler);</pre>	
<b>Parameters (in):</b>	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without RoutineIdentifier)
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction RequestResults request for this DiagnosticRoutine Identifier.	

|(RS\_AP\_00119, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04224)

### 8.3.1.10 CancellationHandler class

[SWS\_DM\_00608][DRAFT] Definition of API class ara::diag::CancellationHandler

|

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/cancellation_handler.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	CancellationHandler
<b>Syntax:</b>	<code>class CancellationHandler final {...};</code>
<b>Description:</b>	CancellationHandler contains a shared state if the processing should be canceled .

|(RS\_Diag\_04169)



### 8.3.1.10.1 diag::CancellationHandler::CancellationHandler function

[SWS\_DM\_00609]{DRAFT} Definition of API function `ara::diag::CancellationHandler::CancellationHandler` [

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

]([RS\\_Diag\\_04169](#), [RS\\_AP\\_00146](#))

[SWS\_DM\_00610]{DRAFT} Definition of API function `ara::diag::CancellationHandler::CancellationHandler` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/cancellation_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::CancellationHandler</a>
<b>Symbol:</b>	CancellationHandler(CancellationHandler &&)
<b>Syntax:</b>	<code>CancellationHandler (CancellationHandler &amp;&amp;) noexcept=default;</code>
<b>DIRECTION NOT DEFINED</b>	CancellationHandler &&      -
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of CancellationHandler.

]([RS\\_AP\\_00133](#), [RS\\_Diag\\_04169](#), [RS\\_AP\\_00145](#))

[SWS\_DM\_00611]{DRAFT} Definition of API function `ara::diag::CancellationHandler::CancellationHandler` [

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

]([RS\\_Diag\\_04169](#), [RS\\_AP\\_00145](#))

[SWS\_DM\_00612]{DRAFT} Definition of API function `ara::diag::CancellationHandler::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/cancellation_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::CancellationHandler</a>
<b>Symbol:</b>	<code>operator=(CancellationHandler &amp;&amp;)</code>



△

<b>Syntax:</b>	<code>CancellationHandler &amp; operator= (CancellationHandler &amp;&amp;)</code> <code>noexcept=default;</code>	
<b>DIRECTION NOT DEFINED</b>	<code>CancellationHandler &amp;&amp;</code>	–
<b>Return value:</b>	<code>CancellationHandler &amp;</code>	–
<b>Exception Safety:</b>	<code>noexcept</code>	
<b>Description:</b>	Move assignment operator of CancellationHandler.	

 ]([RS\\_AP\\_00133](#), [RS\\_Diag\\_04169](#), [RS\\_AP\\_00145](#))

### [SWS\_DM\_00613]{DRAFT} Definition of API function `ara::diag::CancellationHandler::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/cancellation_handler.h"</code>
<b>Scope:</b>	<code>class ara::diag::CancellationHandler</code>
<b>Symbol:</b>	<code>operator=(CancellationHandler &amp;)</code>
<b>Syntax:</b>	<code>CancellationHandler &amp; operator= (CancellationHandler &amp;)=delete;</code>
<b>Description:</b>	CancellationHandler shall be a single not assignable instance.

 ]([RS\\_Diag\\_04169](#), [RS\\_AP\\_00145](#))

### 8.3.1.10.2 `diag::CancellationHandler::IsCanceled` function

### [SWS\_DM\_00614]{DRAFT} Definition of API function `ara::diag::CancellationHandler::IsCanceled` [

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

 ]([RS\\_Diag\\_04169](#))

### 8.3.1.10.3 diag::CancellationHandler::SetNotifier function

[SWS\_DM\_00615]{DRAFT} Definition of API function `ara::diag::CancellationHandler::SetNotifier` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/cancellation_handler.h"	
<b>Scope:</b>	<a href="#">class ara::diag::CancellationHandler</a>	
<b>Symbol:</b>	SetNotifier(std::function< void()> notifier)	
<b>Syntax:</b>	void SetNotifier (std::function< void()> notifier);	
<b>DIRECTION NOT DEFINED</b>	notifier	-
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Registering a notifier function which is called if the diagnostic service execution is canceled in DM. A consecutive call of this method will overwrite the previous registered notifier.	

]([RS\\_Diag\\_04169](#))

### 8.3.1.11 Event class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticEventInterface](#).

[SWS\_DM\_00646]{DRAFT} Definition of API class `ara::diag::Event` [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	Event	
<b>Syntax:</b>	class Event {...};	
<b>Description:</b>	Class to implement operations on diagnostic Events.	

]([RS\\_Diag\\_04151](#))

#### 8.3.1.11.1 diag::Event::EventStatusByte Constructor

[SWS\_DM\_01753]{DRAFT} Definition of API function `ara::diag::EventStatusByte::EventStatusByte` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	EventStatusByte(Args... bits)	



△

<b>Syntax:</b>	<pre>template &lt;typename... Args&gt; constexpr EventStatusByte (Args... bits) noexcept;</pre>	
<b>DIRECTION NOT DEFINED</b>	bits	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Construct an EventStatusByte where all EventStatusBits passed as parameter are set. .	

 ]([RS\\_Diag\\_04151](#))

### 8.3.1.11.2 diag::Event::EventStatusByte Move constructor

[SWS\_DM\_01751]{DRAFT} **Definition of API function ara::diag::EventStatusByte::EventStatusByte** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	EventStatusByte(EventStatusByte &&)	
<b>Syntax:</b>	<pre>constexpr EventStatusByte (EventStatusByte &amp;&amp;) noexcept;</pre>	
<b>DIRECTION NOT DEFINED</b>	EventStatusByte &&	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor .	

 ]([RS\\_Diag\\_04151](#))

### 8.3.1.11.3 diag::Event::EventStatusByte Copy constructor

[SWS\_DM\_01752]{DRAFT} **Definition of API function ara::diag::EventStatusByte::EventStatusByte** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	EventStatusByte(const EventStatusByte &)	
<b>Syntax:</b>	<pre>constexpr EventStatusByte (const EventStatusByte &amp;) noexcept=default;</pre>	
<b>DIRECTION NOT DEFINED</b>	const EventStatusByte &	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Copy constructor .	

 ]([RS\\_Diag\\_04151](#))

### 8.3.1.11.4 diag::Event::EventStatusByte::IsFailedAndTested function

[SWS\_DM\_01757]{DRAFT} Definition of API function ara::diag::EventStatusByte::IsFailedAndTested [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	IsFailedAndTested()	
<b>Syntax:</b>	constexpr bool IsFailedAndTested () const noexcept;	
<b>Return value:</b>	bool	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Whether test is completed this operation cycle and failed. .	

]([RS\\_Diag\\_04151](#))

### 8.3.1.11.5 diag::Event::EventStatusByte::IsNotSet function

[SWS\_DM\_01754]{DRAFT} Definition of API function ara::diag::EventStatusByte::IsNotSet [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	IsNotSet(Args... bits)	
<b>Syntax:</b>	template <typename... Args> constexpr bool IsNotSet (Args... bits) const noexcept;	
<b>DIRECTION NOT DEFINED</b>	bits	–
<b>Return value:</b>	bool	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Whether all EventStatusBits passed as parameter are unset. .	

]([RS\\_Diag\\_04151](#))

### 8.3.1.11.6 diag::Event::EventStatusByte::IsPassedAndTested function

[SWS\_DM\_01756]{DRAFT} Definition of API function ara::diag::EventStatusByte::IsPassedAndTested [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	IsPassedAndTested()	
<b>Syntax:</b>	constexpr bool IsPassedAndTested () const noexcept;	



△

<b>Return value:</b>	bool	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Whether test is completed this operation cycle and passed. .	

](RS\_Diag\_04151)

### 8.3.1.11.7 diag::Event::EventStatusByte::IsSet function

[SWS\_DM\_01755]{DRAFT} Definition of API function ara::diag::EventStatusByte::IsSet [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	IsSet(Args... bits)	
<b>Syntax:</b>	<pre>template &lt;typename... Args&gt; constexpr bool IsSet (Args... bits) const noexcept;</pre>	
<b>DIRECTION NOT DEFINED</b>	bits	–
<b>Return value:</b>	bool	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Whether all EventStatusBits passed as parameter are set. .	

](RS\_Diag\_04151)

### 8.3.1.11.8 diag::Event::EventStatusByte Move assignment operator

[SWS\_DM\_01749]{DRAFT} Definition of API function ara::diag::EventStatusByte::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	operator=(EventStatusByte &&)	
<b>Syntax:</b>	<pre>constexpr EventStatusByte &amp; operator= (EventStatusByte &amp;&amp;) noexcept;</pre>	
<b>DIRECTION NOT DEFINED</b>	EventStatusByte &&	–
<b>Return value:</b>	EventStatusByte &	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assignment operator .	

](RS\_Diag\_04151)

### 8.3.1.11.9 diag::Event::EventStatusByte Copy assignment operator

[SWS\_DM\_01750]{DRAFT} Definition of API function ara::diag::EventStatusByte::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Scope:</b>	<a href="#">struct ara::diag::EventStatusByte</a>	
<b>Symbol:</b>	operator=(const EventStatusByte &)	
<b>Syntax:</b>	constexpr EventStatusByte & operator= (const EventStatusByte &) noexcept;	
<b>DIRECTION NOT DEFINED</b>	const EventStatusByte &	-
<b>Return value:</b>	EventStatusByte &	-
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Copy assignment operator .	

]([RS\\_Diag\\_04151](#))

### 8.3.1.11.10 diag::Event::Event constructor

[SWS\_DM\_00647]{DRAFT} Definition of API function ara::diag::Event::Event [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Event</a>	
<b>Symbol:</b>	Event(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit Event (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor fct. for objects of class Event.	

]([RS\\_Diag\\_04151](#), [RS\\_AP\\_00137](#))

### 8.3.1.11.11 diag::Event::Event(Event&&) move constructor

[SWS\_DM\_01682]{DRAFT} Definition of API function ara::diag::Event::Event [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Event</a>	
<b>Symbol:</b>	Event(Event &&)	
<b>Syntax:</b>	Event (Event &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of Event.	

]([RS\\_AP\\_00147](#))

### 8.3.1.11.12 diag::Event::Event(Event&) copy constructor

[SWS\_DM\_01681]{DRAFT} Definition of API function ara::diag::Event::Event [

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

](RS\_AP\_00147)

### 8.3.1.11.13 diag::Event::Event& operator=(Event&&) move operator

[SWS\_DM\_01680]{DRAFT} Definition of API function ara::diag::Event::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/event.h"
<b>Scope:</b>	class ara::diag::Event
<b>Symbol:</b>	operator=(Event &&)
<b>Syntax:</b>	Event & operator= (Event &&)=delete;
<b>Description:</b>	Move assignment operator of Event.

](RS\_AP\_00147)

### 8.3.1.11.14 diag::Event::Event& operator=(Event&) copy operator

[SWS\_DM\_01679]{DRAFT} Definition of API function ara::diag::Event::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/event.h"
<b>Scope:</b>	class ara::diag::Event
<b>Symbol:</b>	operator=(Event &)
<b>Syntax:</b>	Event & operator= (Event &)=delete;
<b>Description:</b>	Event shall be a single not assignable instance.

](RS\_AP\_00147)



### 8.3.1.11.15 diag::Event::~~Event destructor

[SWS\_DM\_00648]{DRAFT} Definition of API function ara::diag::Event::~~Event [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/event.h"
<b>Scope:</b>	class ara::diag::Event
<b>Symbol:</b>	~Event()
<b>Syntax:</b>	~Event () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class Event.

]([RS\\_Diag\\_04151](#), [RS\\_AP\\_00134](#))

### 8.3.1.11.16 diag::Event::GetEventStatus function

[SWS\_DM\_00649]{DRAFT} Definition of API function ara::diag::Event::GetEventStatus [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/event.h"
<b>Scope:</b>	class ara::diag::Event
<b>Symbol:</b>	GetEventStatus()
<b>Syntax:</b>	ara::core::Future< <a href="#">EventStatusByte</a> > GetEventStatus ();
<b>Return value:</b>	ara::core::Future< Event StatusByte > the current diagnostic event status
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Returns the current diagnostic event status.

]([RS\\_Diag\\_04151](#), [RS\\_AP\\_00139](#))

### 8.3.1.11.17 diag::Event::SetEventStatusChangedNotifier function

[SWS\_DM\_00650]{DRAFT} Definition of API function ara::diag::Event::SetEventStatusChangedNotifier [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/event.h"
<b>Scope:</b>	class ara::diag::Event
<b>Symbol:</b>	SetEventStatusChangedNotifier(std::function< void(EventStatusByte)> notifier)
<b>Syntax:</b>	ara::core::Result< void > SetEventStatusChangedNotifier (std::function< void( <a href="#">EventStatusByte</a> )> notifier);
<b>Parameters (in):</b>	notifier The function to be called if a diagnostic event is changed.
<b>Return value:</b>	ara::core::Result< void > -
<b>Exception Safety:</b>	not exception safe





<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument	given argument is invalid (pointer).
<b>Description:</b>	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.	

|(RS\_Diag\_04183, RS\_AP\_00139)

### 8.3.1.11.18 diag::Event::GetLatchedWIRStatus function

[SWS\_DM\_00651]{DRAFT} Definition of API function ara::diag::Event::GetLatchedWIRStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	GetLatchedWIRStatus()	
<b>Syntax:</b>	ara::core::Future< bool > GetLatchedWIRStatus ();	
<b>Return value:</b>	ara::core::Future< bool >	the current warning indicator status
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Returns the current warning indicator status.	

|(RS\_Diag\_04204, RS\_AP\_00139)

### 8.3.1.11.19 diag::Event::SetLatchedWIRStatus function

[SWS\_DM\_00652]{DRAFT} Definition of API function ara::diag::Event::SetLatchedWIRStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	SetLatchedWIRStatus(bool status)	
<b>Syntax:</b>	ara::core::Result< void > SetLatchedWIRStatus (bool status);	
<b>Parameters (in):</b>	status	Limp-home status as determined by the AA. '0' means limp-home not active; '1' means limp-home active;
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Set the warning indicator status.	

|(RS\_Diag\_04151, RS\_AP\_00139)

### 8.3.1.11.20 diag::Event::GetDTCNumber function

#### [SWS\_DM\_00653]{DRAFT} Definition of API function ara::diag::Event::GetDTC-Number

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	GetDTCNumber(DTCFormatType dtcFormat)	
<b>Syntax:</b>	ara::core::Future< std::uint32_t > GetDTCNumber (DTCFormatType dtcFormat);	
<b>Parameters (in):</b>	dtcFormat	Define DTC format for the return value.
<b>Return value:</b>	ara::core::Future< std::uint32_t >	DTC number in respective DTCFormatType
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kNoSuchDTC	No DTC available.
<b>Description:</b>	Returns the DTC-ID related to this event instance.	

]([RS\\_Diag\\_04201](#), [RS\\_AP\\_00139](#), [RS\\_AP\\_00119](#))

### 8.3.1.11.21 diag::Event::GetDebouncingStatus function

#### [SWS\_DM\_00654]{DRAFT} Definition of API function ara::diag::Event::GetDebouncingStatus

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	GetDebouncingStatus()	
<b>Syntax:</b>	ara::core::Future< DebouncingState > GetDebouncingStatus ();	
<b>Return value:</b>	ara::core::Future< DebouncingState >	Return the current debouncing state of this event.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get the current debouncing status .	

]([RS\\_Diag\\_04068](#), [RS\\_Diag\\_04225](#), [RS\\_AP\\_00139](#))

### 8.3.1.11.22 diag::Event::GetTestComplete function

[SWS\_DM\_00655]{DRAFT} Definition of API function ara::diag::Event::GetTestComplete [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	GetTestComplete()	
<b>Syntax:</b>	ara::core::Future< bool > GetTestComplete ();	
<b>Return value:</b>	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.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get the status if the event has matured to test completed (corresponds to FDC = -128 or FDC = 127).	

](RS\_Diag\_04151, RS\_AP\_00139)

### 8.3.1.11.23 diag::Event::GetFaultDetectionCounter function

[SWS\_DM\_00656]{DRAFT} Definition of API function ara::diag::Event::GetFaultDetectionCounter [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/event.h"	
<b>Scope:</b>	class ara::diag::Event	
<b>Symbol:</b>	GetFaultDetectionCounter()	
<b>Syntax:</b>	ara::core::Future< std::int8_t > GetFaultDetectionCounter ();	
<b>Return value:</b>	ara::core::Future< std::int8_t >	current FaultDetectionCounter value.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Returns the current value of Fault Detection Counter of this event.	

](RS\_Diag\_04068, RS\_AP\_00139)

### 8.3.1.12 Event types class

The *InstanceSpecifier* is only compatible with `PortInterface` of `DiagnosticEventInterface`.

### 8.3.1.12.1 diag::DTCFormatType type

[SWS\_DM\_00642]{DRAFT} Definition of API enum ara::diag::DTCFormatType [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DTCFormatType	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class DTCFormatType : std::uint8_t {...};	
<b>Values:</b>	kDTCFormatOBD= 0	SAE_J2012-DA_DTCFormat_00 as defined in ISO 15031-6 specification.
	kDTCFormatUDS= 1	ISO_14229-1_DTCFormat as defined in ISO 14229-1 specification.
	kDTCFormatJ1939= 2	SAE_J1939-73_DTCFormat as defined in SAE J1939-73.
<b>Description:</b>	Represents the type of the DTC format according to ISO 14229-1.	

]([RS\\_Diag\\_04201](#), [RS\\_AP\\_00125](#))

### 8.3.1.12.2 diag::EventStatusBit type

[SWS\_DM\_00643]{DRAFT} Definition of API enum ara::diag::EventStatusBit [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	EventStatusBit	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class EventStatusBit : std::uint8_t {...};	
<b>Values:</b>	kTestFailed	bit 0: TestFailed
	kTestFailedThisOperationCycle	bit 1: TestFailedThisOperationCycle
	kTestNotCompletedThisOperationCycle	bit 6: TestNotCompletedThisOperationCycle
<b>Description:</b>	Single event status bits.	

]([RS\\_Diag\\_04151](#), [RS\\_AP\\_00125](#))

### 8.3.1.12.3 diag::EventStatusByte type

[SWS\_DM\_00644]{DRAFT} Definition of API class `ara::diag::EventStatusByte` [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/event_types.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>EventStatusByte</code>
<b>Syntax:</b>	<code>struct EventStatusByte {...};</code>
<b>Description:</b>	Current event status byte, bit-encoded.

]([RS\\_Diag\\_04151](#))

### 8.3.1.12.4 diag::DebouncingState type

[SWS\_DM\_00645]{DRAFT} Definition of API enum `ara::diag::DebouncingState` [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/event_types.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>DebouncingState</code>	
<b>Underlying type:</b>	<code>std::uint8_t</code>	
<b>Syntax:</b>	<code>enum class DebouncingState : std::uint8_t {...};</code>	
<b>Values:</b>	<code>kNeutral= 0x00</code>	Neutral (corresponds to FDC = 0)
	<code>kTemporarilyDefective= 0x01</code>	Temporarily Defective (corresponds to $0 < \text{FDC} < 127$ )
	<code>kFinallyDefective= 0x02</code>	finally Defective (corresponds to $\text{FDC} = 127$ )
	<code>kTemporarilyHealed= 0x04</code>	temporarily healed (corresponds to $-128 < \text{FDC} < 0$ )
	<code>kFinallyHealed= 0x08</code>	finally healed (corresponds to $\text{FDC} = -128$ )
<b>Description:</b>	Debounce status of event .	

]([RS\\_Diag\\_04068](#), [RS\\_Diag\\_04225](#), [RS\\_AP\\_00125](#))

### 8.3.1.13 MultipleEvent class

The *InstanceSpecifier* is only compatible with `PortInterface` of `DiagnosticEventInterface`.

**[SWS\_DM\_01704]{DRAFT} Definition of API class ara::diag::MultipleEvent [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/multiple_event.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	MultipleEvent
<b>Syntax:</b>	<code>class MultipleEvent {...};</code>
<b>Description:</b>	Class to implement operations on diagnostic Events with the multiple event interfaces.

]([RS\\_Diag\\_04151](#))

**8.3.1.13.1 diag::EventHandleType type**
**[SWS\_DM\_01703]{DRAFT} Definition of API type ara::diag::EventHandleType [**

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/multiple_event.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	EventHandleType
<b>Syntax:</b>	<code>using EventHandleType = std::uint32_t;</code>
<b>Description:</b>	Type to identify an event .

]([SWS\\_DM\\_01735](#))

**8.3.1.13.2 diag::MultipleEvent::MultipleEvent type**
**[SWS\_DM\_01705]{DRAFT} Definition of API function ara::diag::MultipleEvent::MultipleEvent [**

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/multiple_event.h"		
<b>Scope:</b>	<a href="#">class ara::diag::MultipleEvent</a>		
<b>Symbol:</b>	MultipleEvent(const ara::core::InstanceSpecifier &specifier)		
<b>Syntax:</b>	<code>explicit MultipleEvent (const ara::core::InstanceSpecifier &amp;specifier);</code>		
<b>Parameters (in):</b>	<table border="1"> <tr> <td>specifier</td> <td>InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface</td> </tr> </table>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface
specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEvent Interface		
<b>Exception Safety:</b>	not exception safe		
<b>Description:</b>	Constructor fct. for objects of class MultipleEvent.		

]([RS\\_Diag\\_04151](#), [RS\\_AP\\_00137](#))

### 8.3.1.13.3 diag::MultipleEvent::~Destructor type

[SWS\_DM\_01706]{DRAFT} Definition of API function ara::diag::MultipleEvent::~~MultipleEvent [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/multiple_event.h"
<b>Scope:</b>	class ara::diag::MultipleEvent
<b>Symbol:</b>	~MultipleEvent()
<b>Syntax:</b>	~MultipleEvent () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class MultipleEvent.

](RS\_Diag\_04151, RS\_AP\_00134)

### 8.3.1.13.4 diag::MultipleEvent::~GetEventStatus function

[SWS\_DM\_01707]{DRAFT} Definition of API function ara::diag::MultipleEvent::~GetEventStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_event.h"	
<b>Scope:</b>	class ara::diag::MultipleEvent	
<b>Symbol:</b>	GetEventStatus(EventHandleType eventHandle)	
<b>Syntax:</b>	ara::core::Future< EventStatusByte > GetEventStatus (EventHandleType eventHandle);	
<b>DIRECTION NOT DEFINED</b>	eventHandle	–
<b>Return value:</b>	ara::core::Future< Event StatusByte >	the current diagnostic event status
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument	The provided eventHandle is invalid/unknown
<b>Description:</b>	Returns the current status of the requested event.	

](RS\_Diag\_04151, RS\_AP\_00139)

### 8.3.1.13.5 diag::MultipleEvent::~SetEventStatusChangedNotifier function

[SWS\_DM\_01708]{DRAFT} Definition of API function ara::diag::MultipleEvent::~SetEventStatusChangedNotifier [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/multiple_event.h"
<b>Scope:</b>	class ara::diag::MultipleEvent







<b>Symbol:</b>	SetEventStatusChangedNotifier(EventHandleType eventHandle, std::function< void(EventStatus Byte)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetEventStatusChangedNotifier (EventHandleType eventHandle, std::function< void(EventStatusByte)> notifier);	
<b>Parameters (in):</b>	eventHandle	Identifies the requested event
	notifier	The function to be called if a diagnostic event is changed.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument	given argument is invalid (pointer).
	DiagErrorDomain::Diag Errc::kInvalidArgument	The provided eventHandle is invalid/unknown
<b>Description:</b>	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.	

](RS\_Diag\_04183, RS\_AP\_00139)

### 8.3.1.13.6 diag::MultipleEvent::GetDTCNumber function

[SWS\_DM\_01709][DRAFT] Definition of API function ara::diag::MultipleEvent::GetDTCNumber [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_event.h"	
<b>Scope:</b>	class ara::diag::MultipleEvent	
<b>Symbol:</b>	GetDTCNumber(EventHandleType eventHandle, DTCFormatType dtcFormat)	
<b>Syntax:</b>	ara::core::Future< std::uint32_t > GetDTCNumber (EventHandleType eventHandle, DTCFormatType dtcFormat);	
<b>Parameters (in):</b>	eventHandle	Identifies the requested event
	dtcFormat	Define DTC format for the return value.
<b>Return value:</b>	ara::core::Future< std::uint32_t >	DTC number in respective DTCFormatType
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument	The provided eventHandle is invalid/unknown
	DiagErrorDomain::Diag Errc::kNoSuchDTC	No DTC available.
<b>Description:</b>	Returns the DTC-ID related to requested event.	

](RS\_Diag\_04201, RS\_AP\_00139, RS\_AP\_00119)

### 8.3.1.13.7 diag::MultipleEvent::GetDebouncingStatus function

[SWS\_DM\_01710]{DRAFT} Definition of API function ara::diag::MultipleEvent::GetDebouncingStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_event.h"	
<b>Scope:</b>	class ara::diag::MultipleEvent	
<b>Symbol:</b>	GetDebouncingStatus(EventHandleType eventHandle)	
<b>Syntax:</b>	ara::core::Future< DebouncingState > GetDebouncingStatus (EventHandleType eventHandle);	
<b>Parameters (in):</b>	eventHandle	Identifies the requested event
<b>Return value:</b>	ara::core::Future< DebouncingState >	Return the current debouncing state of this event.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided eventHandle is invalid/unknown
<b>Description:</b>	Get the current debouncing status of the requested event.	

]([RS\\_Diag\\_04068](#), [RS\\_Diag\\_04225](#), [RS\\_AP\\_00139](#))

### 8.3.1.13.8 diag::MultipleEvent::GetFaultDetectionCounter function

[SWS\_DM\_01711]{DRAFT} Definition of API function ara::diag::MultipleEvent::GetFaultDetectionCounter [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_event.h"	
<b>Scope:</b>	class ara::diag::MultipleEvent	
<b>Symbol:</b>	GetFaultDetectionCounter(EventHandleType eventHandle)	
<b>Syntax:</b>	ara::core::Future< std::int8_t > GetFaultDetectionCounter (EventHandleType eventHandle);	
<b>Parameters (in):</b>	eventHandle	Identifies the requested event
<b>Return value:</b>	ara::core::Future< std::int8_t >	current FaultDetectionCounter value.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided eventHandle is invalid/unknown
<b>Description:</b>	Returns the current value of Fault Detection Counter of the selected event.	

]([RS\\_AP\\_00139](#))

### 8.3.1.14 DTCInformation class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticDTCInformationInterface](#).

**[SWS\_DM\_00657]{DRAFT} Definition of API class ara::diag::DTCInformation [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DTCInformation
<b>Syntax:</b>	class DTCInformation {...};
<b>Description:</b>	Class to implement operations on DTC informations per configured DiagnosticMemory Destination.

]([RS\\_Diag\\_04150](#), [RS\\_Diag\\_04105](#))

**8.3.1.14.1 diag::ControlDtcStatusType type**
**[SWS\_DM\_00663]{DRAFT} Definition of API enum ara::diag::ControlDtcStatus Type [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	ControlDtcStatusType	
<b>Underlying type:</b>	uint8_t	
<b>Syntax:</b>	enum class ControlDtcStatusType : uint8_t {...};	
<b>Values:</b>	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.
<b>Description:</b>	Type for ControlDTCStatus status as requested by UDS service 0x85 ControlDTCSetting.	

]([RS\\_Diag\\_04159](#))

**8.3.1.14.2 diag::UdsDtcStatusBitType type**
**[SWS\_DM\_00658]{DRAFT} Definition of API enum ara::diag::UdsDtcStatusBit Type [**

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	UdsDtcStatusBitType	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class UdsDtcStatusBitType : std::uint8_t {...};	
<b>Values:</b>	kTestFailed= 0x01	bit 0: TestFailed





	kTestFailedThis OperationCycle= 0x02	bit 1: TestFailedThisOperationCycle
	kPendingDTC= 0x04	bit 2: PendingDTC
	kConfirmedDTC= 0x08	bit 3: ConfirmedDTC
	kTestNotCompleted SinceLastClear= 0x10	bit 4: TestNotCompletedSinceLastClear
	kTestFailedSinceLast Clear= 0x20	bit 5: TestFailedSinceLastClear
	kTestNotCompletedThis OperationCycle= 0x40	bit 6: TestNotCompletedThisOperationCycle
	kWarningIndicator Requested= 0x80	bit 7: WarningIndicatorRequested
<b>Description:</b>	UDS DTC status bits according to ISO 14229-1.	

|(RS\_Diag\_04067, RS\_Diag\_04151)

### 8.3.1.14.3 diag::DtcSuppressionType type

[SWS\_DM\_01294]{DRAFT} Definition of API enum  
 ara::diag::DTCInformation::DtcSuppressionType [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	
<b>Symbol:</b>	DtcSuppressionType	
<b>Underlying type:</b>	-	
<b>Syntax:</b>	enum class DtcSuppressionType {...};	
<b>Values:</b>	kDTCSuppressionOff= 0x00	suppression status off
	kDTCSuppressionOn= 0x01	suppression status on
<b>Description:</b>	Type for DTC Suppression.	

|(RS\_Diag\_04220)

### 8.3.1.14.4 diag::DTCInformation::UdsDtcStatusByteType type

[SWS\_DM\_00659]{DRAFT} Definition of API class  
 ara::diag::DTCInformation::UdsDtcStatusByteType [

<b>Kind:</b>	struct	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	



△

<b>Symbol:</b>	UdsDtcStatusByteType
<b>Syntax:</b>	struct UdsDtcStatusByteType {...};
<b>Description:</b>	Type for UDS DTC status byte.

 ]([RS\\_Diag\\_04067](#), [RS\\_Diag\\_04151](#))

### 8.3.1.14.5 diag::DTCInformation::SnapshotDataIdentifierType type

[SWS\_DM\_00660]{DRAFT} Definition of API class  
 ara::diag::DTCInformation::SnapshotDataIdentifierType [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	SnapshotDataIdentifierType
<b>Syntax:</b>	struct SnapshotDataIdentifierType {...};
<b>Description:</b>	Type for SnapshotDataIdentifierType status.

 ]([RS\\_Diag\\_04205](#))

### 8.3.1.14.6 diag::DTCInformation::SnapshotDataRecordType type

[SWS\_DM\_00661]{DRAFT} Definition of API class  
 ara::diag::DTCInformation::SnapshotDataRecordType [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	SnapshotDataRecordType
<b>Syntax:</b>	struct SnapshotDataRecordType {...};
<b>Description:</b>	Type for SnapshotDataRecordType status.

 ]([RS\\_Diag\\_04205](#))

### 8.3.1.14.7 diag::DTCInformation::SnapshotRecordUpdatedType type

[SWS\_DM\_00662]{DRAFT} Definition of API class  
 ara::diag::DTCInformation::SnapshotRecordUpdatedType [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	class ara::diag::DTCInformation
<b>Symbol:</b>	SnapshotRecordUpdatedType
<b>Syntax:</b>	struct SnapshotRecordUpdatedType {...};
<b>Description:</b>	Type for SnapshotRecordUpdatedType status.

]([RS\\_Diag\\_04205](#))

### 8.3.1.14.8 diag::DTCInformation::DTCInformation function

[SWS\_DM\_00664]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::DTCInformation [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	class ara::diag::DTCInformation
<b>Symbol:</b>	DTCInformation(const ara::core::InstanceSpecifier &specifier)
<b>Syntax:</b>	explicit DTCInformation (const ara::core::InstanceSpecifier &specifier);
<b>Parameters (in):</b>	specifier InstanceSpecifier to an PortPrototype of an Diagnostic DTCInformationInterface
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Constructor for a DTCInformation instance which allows for DTC related operation per DiagnosticMemoryDestination.

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04150](#), [RS\\_Diag\\_04105](#))

### 8.3.1.14.9 diag::DTCInformation::DTCInformation(DTCInformation&&) move constructor

[SWS\_DM\_01614]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::DTCInformation [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	class ara::diag::DTCInformation
<b>Symbol:</b>	DTCInformation(DTCInformation &&)
<b>Syntax:</b>	DTCInformation (DTCInformation &&) noexcept=delete;



△

<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of DTCInformation.

 ]([RS\\_AP\\_00147](#))

### 8.3.1.14.10 diag::DTCInformation::DTCInformation(DTCInformation&) copy constructor

[SWS\_DM\_01613]{DRAFT} Definition of API function ara::diag::DTCInformation::DTCInformation [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	DTCInformation(DTCInformation &)
<b>Syntax:</b>	DTCInformation (DTCInformation &)=delete;
<b>Description:</b>	DTCInformation shall be a single not copy-able instance.

 ]([RS\\_AP\\_00147](#))

### 8.3.1.14.11 diag::DTCInformation::DTCInformation&tor=(DTCInformation&&) move operator

[SWS\_DM\_01612]{DRAFT} Definition of API function ara::diag::DTCInformation::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	operator=(DTCInformation &&)
<b>Syntax:</b>	<a href="#">DTCInformation</a> & operator= ( <a href="#">DTCInformation</a> &&)=delete;
<b>Description:</b>	Move assignment operator of DTCInformation.

 ]([RS\\_AP\\_00147](#))

### 8.3.1.14.12 diag::DTCInformation::DTCInformation& operator=(DTCInformation&) copy operator opera-

[SWS\_DM\_01611]{DRAFT} **Definition of API function**  
 ara::diag::DTCInformation::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	operator=(DTCInformation &)
<b>Syntax:</b>	<a href="#">DTCInformation</a> & operator= ( <a href="#">DTCInformation</a> &)=delete;
<b>Description:</b>	DTCInformation shall be a single not assignable instance.

] ([RS\\_AP\\_00147](#))

### 8.3.1.14.13 diag::DTCInformation::~~DTCInformation function

[SWS\_DM\_00665]{DRAFT} **Definition of API function**  
 ara::diag::DTCInformation::~~DTCInformation [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	~DTCInformation()
<b>Syntax:</b>	~DTCInformation () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class DTCInformation.

] ([RS\\_AP\\_00134](#))

### 8.3.1.14.14 diag::DTCInformation::GetCurrentStatus function

[SWS\_DM\_00666]{DRAFT} **Definition of API function**  
 ara::diag::DTCInformation::GetCurrentStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>	
<b>Symbol:</b>	GetCurrentStatus(std::uint32_t dtc)	
<b>Syntax:</b>	ara::core::Future< <a href="#">UdsDtcStatusByteType</a> > GetCurrentStatus (std::uint32_t dtc);	
<b>Parameters (in):</b>	dtc	DTC identifier for which the status should be retrieved.
<b>Return value:</b>	ara::core::Future< <a href="#">UdsDtcStatusByteType</a> >	the current UDS DTC status byte of the given DTC identifier.





△

<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kNoSuchDtc	given DTC identifier not available.
<b>Description:</b>	Retrieves the current UDS DTC status byte of the given DTC identifier.	

 ]([RS\\_AP\\_00139](#))

### 8.3.1.14.15 diag::DTCInformation::GetDtcSuppression function

[SWS\_DM\_01293]{DRAFT} Definition of API function **ara::diag::DTCInformation::GetDtcSuppression** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class <a href="#">ara::diag::DTCInformation</a>	
<b>Symbol:</b>	GetDtcSuppression(std::uint32_t dtc)	
<b>Syntax:</b>	ara::core::Result< <a href="#">DtcSuppressionType</a> > GetDtcSuppression (std::uint32_t dtc);	
<b>Parameters (in):</b>	dtc	DTC identifier for which the suppression status should be retrieved.
<b>Return value:</b>	ara::core::Result< DtcSuppressionType >	the current status of DTC Suppression.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kWrongDtc	wrong DTC number requested.
<b>Description:</b>	Contains the current status of DTC Suppression.	

 ]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04220](#))

### 8.3.1.14.16 diag::DTCInformation::SetDTCStatusChangedNotifier function

[SWS\_DM\_00667]{DRAFT} Definition of API function **ara::diag::DTCInformation::SetDTCStatusChangedNotifier** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class <a href="#">ara::diag::DTCInformation</a>	
<b>Symbol:</b>	SetDTCStatusChangedNotifier(std::function< void(std::uint32_t dtc, UdsDtcStatusByteType udsStatusByteOld, UdsDtcStatusByteType udsStatusByteNew)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetDTCStatusChangedNotifier (std::function< void(std::uint32_t dtc, <a href="#">UdsDtcStatusByteType</a> udsStatusByteOld, <a href="#">UdsDtcStatusByteType</a> udsStatusByteNew)> notifier);	
<b>Parameters (in):</b>	notifier	The function to be called if a DTC status has changed.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04148](#))

### 8.3.1.14.17 diag::DTCInformation::SetDtcSuppression function

[SWS\_DM\_01292]{DRAFT} Definition of API function  
ara::diag::DTCInformation::SetDtcSuppression [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	
<b>Symbol:</b>	SetDtcSuppression(std::uint32_t dtc, DtcSuppressionType suppressionStatus)	
<b>Syntax:</b>	ara::core::Result< void > SetDtcSuppression (std::uint32_t dtc, <a href="#">DtcSuppressionType</a> suppressionStatus);	
<b>Parameters (in):</b>	dtc	DTC identifier for which the DTC Suppression status should be set.
	suppressionStatus	DTC Suppression status that should be set.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kWrongDtc	wrong DTC number requested.
	DiagErrorDomain::DiagErrc::kSuppressionIgnored	Suppression request ignored.
<b>Description:</b>	Set the status of DTC Suppression.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04220](#))

### 8.3.1.14.18 diag::DTCInformation::SetSnapshotRecordUpdatedNotifier function

[SWS\_DM\_00668]{DRAFT} Definition of API function  
ara::diag::DTCInformation::SetSnapshotRecordUpdatedNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	
<b>Symbol:</b>	SetSnapshotRecordUpdatedNotifier(std::function< void(SnapshotRecordUpdatedType)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetSnapshotRecordUpdatedNotifier (std::function< void( <a href="#">SnapshotRecordUpdatedType</a> )> notifier);	
<b>Parameters (in):</b>	notifier	The function to be called if the SnapshotRecord is changed.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	given argument is invalid (pointer).
<b>Description:</b>	Register a notifier function which is called if the SnapshotRecord is changed. A consecutive call of this method will overwrite the previous registered notifier.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04205](#))

### 8.3.1.14.19 diag::DTCInformation::GetNumberOfStoredEntries function

[SWS\_DM\_00669]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::GetNumberOfStoredEntries [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	
<b>Symbol:</b>	GetNumberOfStoredEntries()	
<b>Syntax:</b>	ara::core::Future< std::uint32_t > GetNumberOfStoredEntries ();	
<b>Return value:</b>	ara::core::Future< std::uint32_t >	Number of currently stored fault memory entries.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Contains the number of currently stored fault memory entries.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04109](#))

### 8.3.1.14.20 diag::DTCInformation::SetNumberOfStoredEntriesNotifier function

[SWS\_DM\_00670]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::SetNumberOfStoredEntriesNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	class ara::diag::DTCInformation	
<b>Symbol:</b>	SetNumberOfStoredEntriesNotifier(std::function< void(std::uint32_t)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetNumberOfStoredEntriesNotifier (std::function< void(std::uint32_t)> notifier);	
<b>Parameters (in):</b>	notifier	The function to be called if the number of entries for this diagnostic event memory instance has changed.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument	given argument is invalid (pointer).
<b>Description:</b>	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.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04109](#))

### 8.3.1.14.21 diag::DTCInformation::Clear function

[SWS\_DM\_00671]{DRAFT} **Definition of API function**  
**ara::diag::DTCInformation::Clear** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>	
<b>Symbol:</b>	Clear(std::uint32_t dtcGroup)	
<b>Syntax:</b>	ara::core::Future< void > Clear (std::uint32_t dtcGroup);	
<b>Parameters (in):</b>	dtcGroup	DTC group to be cleared.
<b>Return value:</b>	ara::core::Future< void >	void or errors
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kBusy	Busy processing.
	DiagErrorDomain::Diag Errc::kFailed	Clear failed.
	DiagErrorDomain::Diag Errc::kMemoryError	Memory error reported.
	DiagErrorDomain::Diag Errc::kWrongDtc	Wrong DTC group passed.
<b>Description:</b>	Method for Clearing a DTC or a group of DTCs.	

] ([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04194](#))

### 8.3.1.14.22 diag::DTCInformation::GetControlDTCStatus function

[SWS\_DM\_00672]{DRAFT} **Definition of API function**  
**ara::diag::DTCInformation::GetControlDTCStatus** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>	
<b>Symbol:</b>	GetControlDTCStatus()	
<b>Syntax:</b>	ara::core::Future< <a href="#">ControlDtcStatusType</a> > GetControlDTCStatus ();	
<b>Return value:</b>	ara::core::Future< ControlDtcStatusType >	The current status of ControlDtcStatus (related to UDS service 0x85)
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Contains the current status of the ControlDTCStatus.	

] ([RS\\_AP\\_00139](#), [RS\\_Diag\\_04159](#))

### 8.3.1.14.23 diag::DTCInformation::SetControlDtcStatusNotifier function

[SWS\_DM\_00673]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::SetControlDtcStatusNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>	
<b>Symbol:</b>	SetControlDtcStatusNotifier(std::function< void(ControlDtcStatusType)> notifier)	
<b>Syntax:</b>	<pre>ara::core::Result&lt; void &gt; SetControlDtcStatusNotifier (std::function&lt; void(ControlDtcStatusType)&gt; notifier);</pre>	
<b>Parameters (in):</b>	notifier	The function to be called if the ControlDTCStatus (related to UDS service 0x85) for this diagnostic memory instance has changed.
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	given argument is invalid (pointer).
<b>Description:</b>	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.	

] ([RS\\_AP\\_00139](#), [RS\\_Diag\\_04159](#))

### 8.3.1.14.24 diag::DTCInformation::EnableControlDtc function

[SWS\_DM\_00674]{DRAFT} Definition of API function  
 ara::diag::DTCInformation::EnableControlDtc [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dtc_information.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>	
<b>Symbol:</b>	EnableControlDtc()	
<b>Syntax:</b>	<pre>ara::core::Result&lt; void &gt; EnableControlDtc ();</pre>	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

] ([RS\\_AP\\_00139](#), [RS\\_Diag\\_04159](#))

### 8.3.1.14.25 diag::DTCInformation::GetEventMemoryOverflow function

[SWS\_DM\_00919]{DRAFT} **Definition of API function**  
**ara::diag::DTCInformation::GetEventMemoryOverflow** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	GetEventMemoryOverflow()
<b>Syntax:</b>	<code>ara::core::Result&lt; bool &gt; GetEventMemoryOverflow ();</code>
<b>Return value:</b>	ara::core::Result< bool >   Current status of event memory overflow.
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Contains the current event memory overflow status.

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04093](#))

### 8.3.1.14.26 diag::DTCInformation::SetEventMemoryOverflowNotifier function

[SWS\_DM\_00918]{DRAFT} **Definition of API function**  
**ara::diag::DTCInformation::SetEventMemoryOverflowNotifier** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dtc_information.h"
<b>Scope:</b>	<a href="#">class ara::diag::DTCInformation</a>
<b>Symbol:</b>	SetEventMemoryOverflowNotifier(std::function< void(bool)> notifier)
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; SetEventMemoryOverflowNotifier (std::function&lt; void(bool)&gt; notifier);</code>
<b>Parameters (in):</b>	notifier   The function to be called if the overflow status for this diagnostic event memory instance has changed.
<b>Return value:</b>	ara::core::Result< void >   –
<b>Exception Safety:</b>	not exception safe
<b>Errors:</b>	DiagErrorDomain::Diag Errc::kInvalidArgument   given argument is invalid (pointer).
<b>Description:</b>	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.

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04093](#))

### 8.3.1.15 Conversation class

The conversation object can only be retrieved by a given meta\_info object.

**[SWS\_DM\_00693]{DRAFT} Definition of API class ara::diag::Conversation [**

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	Conversation
<b>Syntax:</b>	<code>class Conversation {...};</code>
<b>Description:</b>	Conversation interface.

]([RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#))

**8.3.1.15.1 diag::ActivityStatusType type**
**[SWS\_DM\_00690]{DRAFT} Definition of API enum ara::diag::ActivityStatusType [**

<b>Kind:</b>	enumeration				
<b>Header file:</b>	#include "ara/diag/conversation.h"				
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"				
<b>Scope:</b>	namespace ara::diag				
<b>Symbol:</b>	ActivityStatusType				
<b>Underlying type:</b>	std::uint8_t				
<b>Syntax:</b>	<code>enum class ActivityStatusType : std::uint8_t {...};</code>				
<b>Values:</b>	<table border="1"> <tr> <td>kActive= 0x00</td> <td>Currently active; i.e. request is currently processed or non-default session is active.</td> </tr> <tr> <td>kInactive= 0x01</td> <td>Currently not active.</td> </tr> </table>	kActive= 0x00	Currently active; i.e. request is currently processed or non-default session is active.	kInactive= 0x01	Currently not active.
kActive= 0x00	Currently active; i.e. request is currently processed or non-default session is active.				
kInactive= 0x01	Currently not active.				
<b>Description:</b>	Type for current activity status.				

]([RS\\_AP\\_00125](#), [RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04209](#))

**8.3.1.15.2 diag::SessionControlType type**
**[SWS\_DM\_00706]{DRAFT} Definition of API type ara::diag::SessionControlType [**

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SessionControlType
<b>Syntax:</b>	<code>using SessionControlType = std::uint8_t;</code>
<b>Description:</b>	Type for the active diagnostic session. .

]([RS\\_AP\\_00125](#))

### 8.3.1.15.3 diag::SessionControlType kDefaultSession

[SWS\_DM\_01278]{DRAFT} Definition of API variable ara::diag::kDefaultSession [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kDefaultSession
<b>Type:</b>	SessionControlType
<b>Syntax:</b>	constexpr SessionControlType kDefaultSession = 0x01;
<b>Description:</b>	Default session according to ISO 14229-1.

](RS\_AP\_00125)

### 8.3.1.15.4 diag::SessionControlType kExtendedDiagnosticSession

[SWS\_DM\_01280]{DRAFT} Definition of API variable ara::diag::kExtendedDiagnosticSession [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kExtendedDiagnosticSession
<b>Type:</b>	SessionControlType
<b>Syntax:</b>	constexpr SessionControlType kExtendedDiagnosticSession = 0x03;
<b>Description:</b>	Extended diagnostic session according to ISO 14229-1.

](RS\_AP\_00125)

### 8.3.1.15.5 diag::SessionControlType kProgrammingSession

[SWS\_DM\_01279]{DRAFT} Definition of API variable ara::diag::kProgrammingSession [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kProgrammingSession
<b>Type:</b>	SessionControlType
<b>Syntax:</b>	constexpr SessionControlType kProgrammingSession = 0x02;
<b>Description:</b>	Programming session according to ISO 14229-1.

](RS\_AP\_00125)



### 8.3.1.15.6 diag::SessionControlType kSafetySystemDiagnosticSession

[SWS\_DM\_01281]{DRAFT} Definition of API variable ara::diag::kSafetySystemDiagnosticSession [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kSafetySystemDiagnosticSession
<b>Type:</b>	SessionControlType
<b>Syntax:</b>	constexpr SessionControlType kSafetySystemDiagnosticSession = 0x04;
<b>Description:</b>	Safety system diagnostic session according to ISO 14229-1.

](RS\_AP\_00125)

### 8.3.1.15.7 diag::SecurityLevelType type

[SWS\_DM\_00705]{DRAFT} Definition of API type ara::diag::SecurityLevelType [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SecurityLevelType
<b>Syntax:</b>	using SecurityLevelType = std::uint8_t;
<b>Description:</b>	Type for the active security level. .

](RS\_AP\_00125)

### 8.3.1.15.8 diag::SecurityLevelType kLocked

[SWS\_DM\_01282]{DRAFT} Definition of API variable ara::diag::kLocked [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kLocked
<b>Type:</b>	SecurityLevelType
<b>Syntax:</b>	constexpr SecurityLevelType kLocked = 0x00;
<b>Description:</b>	Security level locked.

](RS\_AP\_00125)

### 8.3.1.15.9 diag::Conversation::ConversationIdentifierType type

[SWS\_DM\_00691]{DRAFT} Definition of API class  
 ara::diag::Conversation::ConversationIdentifierType [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/conversation.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>
<b>Symbol:</b>	ConversationIdentifierType
<b>Syntax:</b>	struct ConversationIdentifierType {...};
<b>Description:</b>	Properties allowing an identification of the conversation.

]([RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04209](#))

### 8.3.1.15.10 diag::Conversation::GetConversation function

[SWS\_DM\_00692]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetConversation [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetConversation(MetaInfo &metaInfo)	
<b>Syntax:</b>	static ara::core::Result< <a href="#">Conversation</a> & > GetConversation ( <a href="#">MetaInfo</a> &metaInfo);	
<b>Parameters (in):</b>	metaInfo	contains additional meta information
<b>Return value:</b>	ara::core::Result< <a href="#">Conversation</a> & >	Conversation object or error
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get one conversation based on given MetaInfo.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04170](#))

### 8.3.1.15.11 diag::Conversation::GetAllConversations function

[SWS\_DM\_00782]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetAllConversations [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetAllConversations()	
<b>Syntax:</b>	static ara::core::Vector< std::reference_wrapper< <a href="#">Conversation</a> > > GetAllConversations ();	



△

<b>Return value:</b>	ara::core::Vector<std::reference_wrapper<Conversation >>	a vector of all possible Conversation objects
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get all possible conversations.	

]([RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04209](#))

### 8.3.1.15.12 diag::Conversation::GetCurrentActiveConversations function

[SWS\_DM\_00783]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetCurrentActiveConversations [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetCurrentActiveConversations()	
<b>Syntax:</b>	static ara::core::Vector<std::reference_wrapper< <a href="#">Conversation</a> >> GetCurrentActiveConversations ();	
<b>Return value:</b>	ara::core::Vector<std::reference_wrapper<Conversation >>	a vector of all currently active (GetActivityStatus() == kActive) Conversation objects
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get all currently active conversations.	

]([RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04209](#))

### 8.3.1.15.13 diag::Conversation::GetActivityStatus function

[SWS\_DM\_00694]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetActivityStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetActivityStatus()	
<b>Syntax:</b>	ara::core::Result< <a href="#">ActivityStatusType</a> > GetActivityStatus ();	
<b>Return value:</b>	ara::core::Result<ActivityStatusType >	the activity status of the conversation
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Represents the status of an active conversation.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.1.15.14 diag::Conversation::SetActivityNotifier function

[SWS\_DM\_00695]{DRAFT} Definition of API function  
ara::diag::Conversation::SetActivityNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	SetActivityNotifier(std::function< void(ActivityStatusType)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetActivityNotifier (std::function< void(ActivityStatusType)> notifier);	
<b>Parameters (in):</b>	notifier	notifier function to be called
<b>Return value:</b>	ara::core::Result< void >	void when the registering went fine or error
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Register a notifier function which is called if the activity is changed. A consecutive call of this method will overwrite the previous registered notifier.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.1.15.15 diag::Conversation::GetConversationIdentifier function

[SWS\_DM\_00700]{DRAFT} Definition of API function  
ara::diag::Conversation::GetConversationIdentifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetConversationIdentifier()	
<b>Syntax:</b>	ara::core::Result< <a href="#">ConversationIdentifierType</a> > GetConversationIdentifier ();	
<b>Return value:</b>	ara::core::Result< ConversationIdentifierType >	the conversation information
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Getter for the current identification properties of the active conversation.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.1.15.16 diag::Conversation::GetDiagnosticSession function

[SWS\_DM\_00696]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetDiagnosticSession [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetDiagnosticSession()	
<b>Syntax:</b>	ara::core::Result< <a href="#">SessionControlType</a> > GetDiagnosticSession ();	
<b>Return value:</b>	ara::core::Result< SessionControlType >	the current session
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Represents the current active diagnostic session of an active conversation.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04208](#))

### 8.3.1.15.17 diag::Conversation::SetDiagnosticSessionNotifier function

[SWS\_DM\_00697]{DRAFT} Definition of API function  
 ara::diag::Conversation::SetDiagnosticSessionNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	SetDiagnosticSessionNotifier(std::function< void(SessionControlType)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetDiagnosticSessionNotifier (std::function< void( <a href="#">SessionControlType</a> )> notifier);	
<b>Parameters (in):</b>	notifier	notifier function to be called
<b>Return value:</b>	ara::core::Result< void >	void when the registering went fine or error
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Register a notifier function which is called if the Session is changed. A consecutive call of this method will overwrite the previous registered notifier.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04208](#))

### 8.3.1.15.18 diag::Conversation::GetDiagnosticSessionShortName function

[SWS\_DM\_00707]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetDiagnosticSessionShortName [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Conversation</a>	
<b>Symbol:</b>	GetDiagnosticSessionShortName(SessionControlType session)	





<b>Syntax:</b>	ara::core::Result< ara::core::StringView > GetDiagnosticSessionShortName (SessionControlType session);	
<b>Parameters (in):</b>	session	Diagnostic session the shortname shall be returned for.
<b>Return value:</b>	ara::core::Result< ara::core::StringView >	ara::core::Result<ara::core::StringView> the session as short Name; DiagnosticSession.shortName
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The requested session is invalid
<b>Description:</b>	Converts the given diagnostic session into the ShortName.	

](RS\_Diag\_04169, RS\_Diag\_04208)

### 8.3.1.15.19 diag::Conversation::GetDiagnosticSecurityLevel function

[SWS\_DM\_00698]{DRAFT} Definition of API function  
 ara::diag::Conversation::GetDiagnosticSecurityLevel [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	class ara::diag::Conversation	
<b>Symbol:</b>	GetDiagnosticSecurityLevel()	
<b>Syntax:</b>	ara::core::Result< SecurityLevelType > GetDiagnosticSecurityLevel ();	
<b>Return value:</b>	ara::core::Result< SecurityLevelType >	the current SecurityLevel
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Represents the current active diagnostic SecurityLevel of an active conversation.	

](RS\_AP\_00139, RS\_Diag\_04169, RS\_Diag\_04208)

### 8.3.1.15.20 diag::Conversation::SetSecurityLevelNotifier function

[SWS\_DM\_00699]{DRAFT} Definition of API function  
 ara::diag::Conversation::SetSecurityLevelNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	class ara::diag::Conversation	
<b>Symbol:</b>	SetSecurityLevelNotifier(std::function< void(SecurityLevelType)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetSecurityLevelNotifier (std::function< void(SecurityLevelType)> notifier);	
<b>Parameters (in):</b>	notifier	notifier function to be called
<b>Return value:</b>	ara::core::Result< void >	void when the registering went fine or error
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Register a notifier function which is called if the SecurityLevel is changed. A consecutive call of this method will overwrite the previous registered notifier.	

](RS\_AP\_00139, RS\_Diag\_04169, RS\_Diag\_04208)

### 8.3.1.15.21 diag::Conversation::GetDiagnosticSecurityLevelShortName function

[SWS\_DM\_00708]{DRAFT} Definition of API function ara::diag::Conversation::GetDiagnosticSecurityLevelShortName [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	class ara::diag::Conversation	
<b>Symbol:</b>	GetDiagnosticSecurityLevelShortName(SecurityLevelType securityLevel)	
<b>Syntax:</b>	ara::core::Result< ara::core::StringView > GetDiagnosticSecurityLevelShortName (SecurityLevelType securityLevel);	
<b>Parameters (in):</b>	securityLevel	Security level enum the shortname shall be returned for.
<b>Return value:</b>	ara::core::Result< ara::core::StringView >	ara::core::Result<ara::core::StringView> the SecurityLevel as short Name; DiagnosticSecurityLevel.shortName
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The requested security level is invalid
<b>Description:</b>	Converts the given diagnostic SecurityLevel into the ShortName.	

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04208](#))

### 8.3.1.15.22 diag::Conversation::ResetToDefaultSession function

[SWS\_DM\_00701]{DRAFT} Definition of API function ara::diag::Conversation::ResetToDefaultSession [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/conversation.h"	
<b>Scope:</b>	class ara::diag::Conversation	
<b>Symbol:</b>	ResetToDefaultSession()	
<b>Syntax:</b>	ara::core::Result< void > ResetToDefaultSession ();	
<b>Return value:</b>	ara::core::Result< void >	void on success or error
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Method to reset the current session to default session.	

]([RS\\_Diag\\_04006](#), [RS\\_Diag\\_04166](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04209](#))

### 8.3.1.16 Condition class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticConditionInterface](#).

[SWS\_DM\_00711]{DRAFT} Definition of API class `ara::diag::Condition` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	Condition
<b>Syntax:</b>	<code>class Condition {...};</code>
<b>Description:</b>	DiagnosticConditionInterface.

]([RS\\_Diag\\_04192](#))

### 8.3.1.16.1 `diag::ConditionType` type

[SWS\_DM\_00710]{DRAFT} Definition of API enum `ara::diag::ConditionType` [

<b>Kind:</b>	enumeration				
<b>Header file:</b>	#include "ara/diag/condition.h"				
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"				
<b>Scope:</b>	namespace <code>ara::diag</code>				
<b>Symbol:</b>	ConditionType				
<b>Underlying type:</b>	–				
<b>Syntax:</b>	<code>enum class ConditionType {...};</code>				
<b>Values:</b>	<table border="1"> <tr> <td><code>kConditionFalse= 0x00</code></td> <td>condition is set to false</td> </tr> <tr> <td><code>kConditionTrue= 0x01</code></td> <td>condition is set to true</td> </tr> </table>	<code>kConditionFalse= 0x00</code>	condition is set to false	<code>kConditionTrue= 0x01</code>	condition is set to true
<code>kConditionFalse= 0x00</code>	condition is set to false				
<code>kConditionTrue= 0x01</code>	condition is set to true				
<b>Description:</b>	Type for Condition status.				

]([RS\\_AP\\_00125](#), [RS\\_Diag\\_04192](#))

### 8.3.1.16.2 `diag::Condition::Condition` function

[SWS\_DM\_00712]{DRAFT} Definition of API function `ara::diag::Condition::Condition` [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/condition.h"		
<b>Scope:</b>	<a href="#">class <code>ara::diag::Condition</code></a>		
<b>Symbol:</b>	<code>Condition(const ara::core::InstanceSpecifier &amp;specifier)</code>		
<b>Syntax:</b>	<code>explicit Condition (const ara::core::InstanceSpecifier &amp;specifier);</code>		
<b>Parameters (in):</b>	<table border="1"> <tr> <td>specifier</td> <td>InstanceSpecifier to an PortPrototype of an DiagnosticCondition Interface</td> </tr> </table>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticCondition Interface
specifier	InstanceSpecifier to an PortPrototype of an DiagnosticCondition Interface		
<b>Exception Safety:</b>	not exception safe		
<b>Description:</b>	Constructor of Condition Class.		

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04192](#))



### 8.3.1.16.3 diag::Condition::Condition(Condition&&) move constructor

[SWS\_DM\_01626]{DRAFT} Definition of API function  
 ara::diag::Condition::Condition [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>
<b>Symbol:</b>	Condition(Condition &&)
<b>Syntax:</b>	Condition (Condition &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of Condition.

]([RS\\_AP\\_00147](#))

### 8.3.1.16.4 diag::Condition::Condition(Condition&) copy constructor

[SWS\_DM\_01625]{DRAFT} Definition of API function  
 ara::diag::Condition::Condition [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>
<b>Symbol:</b>	Condition(Condition &)
<b>Syntax:</b>	Condition (Condition &)=delete;
<b>Description:</b>	Condition shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.16.5 diag::Condition::Condition& operator=(Condition&&) move operator

[SWS\_DM\_01624]{DRAFT} Definition of API function  
 ara::diag::Condition::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>
<b>Symbol:</b>	operator=(Condition &&)
<b>Syntax:</b>	<a href="#">Condition</a> & operator= ( <a href="#">Condition</a> &&)=delete;
<b>Description:</b>	Move assignment operator of Condition.

]([RS\\_AP\\_00147](#))

### 8.3.1.16.6 diag::Condition::Condition& operator=(Condition&) copy operator

[SWS\_DM\_01623]{DRAFT} Definition of API function  
 ara::diag::Condition::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>
<b>Symbol:</b>	operator=(Condition &)
<b>Syntax:</b>	<a href="#">Condition</a> & operator= ( <a href="#">Condition</a> &)=delete;
<b>Description:</b>	Condition shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.16.7 diag::Condition::~~Condition function

[SWS\_DM\_00713]{DRAFT} Definition of API function  
 ara::diag::Condition::~~Condition [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/condition.h"
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>
<b>Symbol:</b>	~Condition()
<b>Syntax:</b>	~Condition () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class Condition.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04192](#))

### 8.3.1.16.8 diag::Condition::GetCondition function

[SWS\_DM\_00714]{DRAFT} Definition of API function ara::diag::Condition::GetCondition [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/condition.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>	
<b>Symbol:</b>	GetCondition()	
<b>Syntax:</b>	ara::core::Future< <a href="#">ConditionType</a> > GetCondition ();	
<b>Return value:</b>	ara::core::Future< <a href="#">ConditionType</a> >	the current condition
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get current condition.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04192](#))

### 8.3.1.16.9 diag::Condition::SetCondition function

[SWS\_DM\_00715]{DRAFT} Definition of API function `ara::diag::Condition::SetCondition` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/condition.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Condition</a>	
<b>Symbol:</b>	SetCondition(ConditionType condition)	
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; SetCondition (ConditionType condition);</code>	
<b>Parameters (in):</b>	condition	current condition
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Set condition.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04192](#))

### 8.3.1.17 MultipleCondition class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticConditionInterface](#).

[SWS\_DM\_01730]{DRAFT} Definition of API class `ara::diag::MultipleCondition` [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	MultipleCondition	
<b>Syntax:</b>	<code>class MultipleCondition {...};</code>	
<b>Description:</b>	Class for multiple condition operations.	

]([RS\\_Diag\\_04192](#))

### 8.3.1.17.1 diag::ConditionHandleType type

[SWS\_DM\_01726]{DRAFT} Definition of API type `ara::diag::ConditionHandleType` [

<b>Kind:</b>	type alias	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	ConditionHandleType	
<b>Syntax:</b>	<code>using ConditionHandleType = std::uint8_t;</code>	
<b>Description:</b>	Type to identify a condition .	

]([RS\\_Diag\\_04192](#))

### 8.3.1.17.2 diag::MultipleCondition::MultipleCondition type

[SWS\_DM\_01727]{DRAFT} Definition of API function ara::diag::MultipleCondition::MultipleCondition [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Scope:</b>	class ara::diag::MultipleCondition	
<b>Symbol:</b>	MultipleCondition(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit MultipleCondition (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMultipleConditionInterface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of MultipleCondition Class.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04192](#))

### 8.3.1.17.3 diag::MultipleCondition::~Destructor type

[SWS\_DM\_01737]{DRAFT} Definition of API function ara::diag::MultipleCondition::~~MultipleCondition [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Scope:</b>	class ara::diag::MultipleCondition	
<b>Symbol:</b>	~MultipleCondition()	
<b>Syntax:</b>	~MultipleCondition () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructor of class Condition.	

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04192](#))

### 8.3.1.17.4 diag::MultipleCondition::GetCondition function

[SWS\_DM\_01728]{DRAFT} Definition of API function ara::diag::MultipleCondition::GetCondition [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Scope:</b>	class ara::diag::MultipleCondition	
<b>Symbol:</b>	GetCondition(ConditionHandleType conditionHandle)	
<b>Syntax:</b>	ara::core::Future< ConditionType > GetCondition (ConditionHandleType conditionHandle);	
<b>Parameters (in):</b>	conditionHandle	Identifies the requested condition





<b>Return value:</b>	ara::core::Future<ConditionType >	the current condition
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided conditionHandle is invalid/unkwnown
<b>Description:</b>	Get condition state of the requested condition.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04192](#))

### 8.3.1.17.5 diag::MultipleCondition::SetCondition function

[SWS\_DM\_01729][DRAFT] Definition of API function ara::diag::MultipleCondition::SetCondition [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/multiple_condition.h"	
<b>Scope:</b>	<a href="#">class ara::diag::MultipleCondition</a>	
<b>Symbol:</b>	SetCondition(ConditionHandleType conditionHandle, ConditionType condition)	
<b>Syntax:</b>	ara::core::Result< void > SetCondition ( <a href="#">ConditionHandleType</a> conditionHandle, <a href="#">ConditionType</a> condition);	
<b>Parameters (in):</b>	conditionHandle	Identifies the requested condition
	condition	current condition
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagErrc::kInvalidArgument	The provided conditionHandle is invalid/unkwnown
<b>Description:</b>	Set condition state of the requested condition.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04192](#))

### 8.3.1.18 OperationCycle class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticOperationCycleInterface](#).

[SWS\_DM\_00751][DRAFT] Definition of API class ara::diag::OperationCycle [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	OperationCycle	
<b>Syntax:</b>	class OperationCycle {...};	
<b>Description:</b>	DiagnosticOperationCycleInterface provides functionality for handling of operation cycles.	

]([RS\\_Diag\\_04178](#))

### 8.3.1.18.1 diag::OperationCycle::OperationCycle function

[SWS\_DM\_00752]{DRAFT} Definition of API function ara::diag::OperationCycle::OperationCycle [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"	
<b>Scope:</b>	<a href="#">class ara::diag::OperationCycle</a>	
<b>Symbol:</b>	OperationCycle(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit OperationCycle (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticOperation CycleInterface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor for DiagnosticOperationCycleInterface.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04178](#))

### 8.3.1.18.2 diag::OperationCycle::OperationCycle(OperationCycle&&) move constructor

[SWS\_DM\_01599]{DRAFT} Definition of API function ara::diag::OperationCycle::OperationCycle [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"	
<b>Scope:</b>	<a href="#">class ara::diag::OperationCycle</a>	
<b>Symbol:</b>	OperationCycle(OperationCycle &&)	
<b>Syntax:</b>	OperationCycle (OperationCycle &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of OperationCycle.	

]([RS\\_AP\\_00147](#))

### 8.3.1.18.3 diag::OperationCycle::OperationCycle(OperationCycle&) copy constructor

[SWS\_DM\_01598]{DRAFT} Definition of API function ara::diag::OperationCycle::OperationCycle [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"	
<b>Scope:</b>	<a href="#">class ara::diag::OperationCycle</a>	
<b>Symbol:</b>	OperationCycle(OperationCycle &)	



△

<b>Syntax:</b>	<code>OperationCycle (OperationCycle &amp;)=delete;</code>
<b>Description:</b>	OperationCycle shall be a single not copy-able instance.

](RS\_AP\_00147)

#### 8.3.1.18.4 diag::OperationCycle::OperationCycle& operator=(OperationCycle&&) move operator

[SWS\_DM\_01597][DRAFT] Definition of API function ara::diag::OperationCycle::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"
<b>Scope:</b>	class ara::diag::OperationCycle
<b>Symbol:</b>	operator=(OperationCycle &&)
<b>Syntax:</b>	<code>OperationCycle &amp; operator= (OperationCycle &amp;&amp;)=delete;</code>
<b>Description:</b>	Move assignment operator of OperationCycle.

](RS\_AP\_00147)

#### 8.3.1.18.5 diag::OperationCycle::OperationCycle& operator=(OperationCycle&) copy operator

[SWS\_DM\_01596][DRAFT] Definition of API function ara::diag::OperationCycle::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"
<b>Scope:</b>	class ara::diag::OperationCycle
<b>Symbol:</b>	operator=(OperationCycle &)
<b>Syntax:</b>	<code>OperationCycle &amp; operator= (OperationCycle &amp;)=delete;</code>
<b>Description:</b>	OperationCycle shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.18.6 diag::OperationCycle::~~OperationCycle function

[SWS\_DM\_00753]{DRAFT} Definition of API function ara::diag::OperationCycle::~~OperationCycle [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"
<b>Scope:</b>	class ara::diag::OperationCycle
<b>Symbol:</b>	~OperationCycle()
<b>Syntax:</b>	~OperationCycle () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DiagnosticOperationCycleInterface.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04178](#))

### 8.3.1.18.7 diag::OperationCycle::SetNotifier function

[SWS\_DM\_00755]{DRAFT} Definition of API function ara::diag::OperationCycle::SetNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"	
<b>Scope:</b>	class ara::diag::OperationCycle	
<b>Symbol:</b>	SetNotifier(std::function< void()> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetNotifier (std::function< void()> notifier);	
<b>DIRECTION NOT DEFINED</b>	notifier	–
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04178](#), [RS\\_Diag\\_04186](#))

### 8.3.1.18.8 diag::OperationCycle::RestartOperationCycle function

[SWS\_DM\_01102]{DRAFT} Definition of API function ara::diag::OperationCycle::RestartOperationCycle [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/operation_cycle.h"
<b>Scope:</b>	class ara::diag::OperationCycle
<b>Symbol:</b>	RestartOperationCycle()
<b>Syntax:</b>	ara::core::Result< void > RestartOperationCycle ();







<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Trigger to restart the OperationCycle.	

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04178](#), [RS\\_Diag\\_04182](#))

### 8.3.1.19 Indicator class

The *InstanceSpecifier* is only compatible with *PortInterface* of *DiagnosticIndicatorInterface*.

#### [SWS\_DM\_00741]{DRAFT} Definition of API class ara::diag::Indicator [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/indicator.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	Indicator	
<b>Syntax:</b>	class Indicator {...};	
<b>Description:</b>	DiagnosticIndicatorInterface provides functionality for handling indicators.	

]([RS\\_Diag\\_04204](#))

#### 8.3.1.19.1 diag::IndicatorType type

#### [SWS\_DM\_00740]{DRAFT} Definition of API enum ara::diag::IndicatorType [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/indicator.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	IndicatorType	
<b>Underlying type:</b>	–	
<b>Syntax:</b>	enum class IndicatorType {...};	
<b>Values:</b>	kOff= 0x00	Indicator off mode {default}.
	kContinuous= 0x01	Indicator continuously on mode.
	kBlinking= 0x02	Indicator blinking mode.
	kBlinkingOrContinuous= 0x03	Indicator blinking Or continuously on mode.
	kSlowFlash= 0x04	Indicator slow flashing mode.
	kFastFlash= 0x05	Indicator fast flashing mode.
	kOnDemand= 0x06	Indicator on-demand mode.
	kShort= 0x07	Indicator short mode.





<b>Description:</b>	Represents the state of an indicator.
---------------------	---------------------------------------

]([RS\\_Diag\\_04204](#))

### 8.3.1.19.2 diag::Indicator::Indicator function

[SWS\_DM\_00742]{DRAFT} **Definition of API function**  
**ara::diag::Indicator::Indicator** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/indicator.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>	
<b>Symbol:</b>	Indicator(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit Indicator (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticIndicator Interface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor for DiagnosticIndicatorInterface.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04204](#))

### 8.3.1.19.3 diag::Indicator::Indicator(Indicator&&) move constructor

[SWS\_DM\_01630]{DRAFT} **Definition of API function**  
**ara::diag::Indicator::Indicator** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/indicator.h"	
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>	
<b>Symbol:</b>	Indicator(Indicator &&)	
<b>Syntax:</b>	Indicator (Indicator &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of Indicator.	

]([RS\\_AP\\_00147](#))

### 8.3.1.19.4 diag::Indicator::Indicator(Indicator&) copy constructor

[SWS\_DM\_01629]{DRAFT} Definition of API function  
 ara::diag::Indicator::Indicator [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/indicator.h"
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>
<b>Symbol:</b>	Indicator(Indicator &)
<b>Syntax:</b>	Indicator (Indicator &)=delete;
<b>Description:</b>	Indicator shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.19.5 diag::Indicator::Indicator& operator=(Indicator&&) move operator

[SWS\_DM\_01628]{DRAFT} Definition of API function  
 ara::diag::Indicator::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/indicator.h"
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>
<b>Symbol:</b>	operator=(Indicator &&)
<b>Syntax:</b>	<a href="#">Indicator</a> & operator= ( <a href="#">Indicator</a> &&)=delete;
<b>Description:</b>	Move assignment operator of Indicator.

]([RS\\_AP\\_00147](#))

### 8.3.1.19.6 diag::Indicator::Indicator& operator=(Indicator&) copy operator

[SWS\_DM\_01627]{DRAFT} Definition of API function  
 ara::diag::Indicator::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/indicator.h"
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>
<b>Symbol:</b>	operator=(Indicator &)
<b>Syntax:</b>	<a href="#">Indicator</a> & operator= ( <a href="#">Indicator</a> &)=delete;
<b>Description:</b>	Indicator shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.19.7 diag::Indicator::~~Indicator function

[SWS\_DM\_00743]{DRAFT} **Definition of API function** `ara::diag::Indicator::~~Indicator` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/indicator.h"
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>
<b>Symbol:</b>	~Indicator()
<b>Syntax:</b>	~Indicator () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DiagnosticIndicatorInterface.

] ([RS\\_AP\\_00134](#), [RS\\_Diag\\_04204](#))

### 8.3.1.19.8 diag::Indicator::GetIndicator function

[SWS\_DM\_00744]{DRAFT} **Definition of API function** `ara::diag::Indicator::GetIndicator` [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/indicator.h"		
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>		
<b>Symbol:</b>	GetIndicator()		
<b>Syntax:</b>	<code>ara::core::Future&lt; <a href="#">IndicatorType</a> &gt; GetIndicator ();</code>		
<b>Return value:</b>	<table border="1"> <tr> <td><code>ara::core::Future&lt; <a href="#">IndicatorType</a> &gt;</code></td> <td>the current Indicator</td> </tr> </table>	<code>ara::core::Future&lt; <a href="#">IndicatorType</a> &gt;</code>	the current Indicator
<code>ara::core::Future&lt; <a href="#">IndicatorType</a> &gt;</code>	the current Indicator		
<b>Exception Safety:</b>	not exception safe		
<b>Description:</b>	Get current Indicator.		

] ([RS\\_AP\\_00139](#), [RS\\_Diag\\_04204](#))

### 8.3.1.19.9 diag::Indicator::SetNotifier function

[SWS\_DM\_00745]{DRAFT} **Definition of API function** `ara::diag::Indicator::SetNotifier` [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/indicator.h"		
<b>Scope:</b>	<a href="#">class ara::diag::Indicator</a>		
<b>Symbol:</b>	SetNotifier(std::function< void(IndicatorType)> notifier)		
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; SetNotifier (std::function&lt; void(<a href="#">IndicatorType</a>)&gt; notifier);</code>		
<b>Parameters (in):</b>	<table border="1"> <tr> <td>notifier</td> <td>notifier function</td> </tr> </table>	notifier	notifier function
notifier	notifier function		
<b>Return value:</b>	<table border="1"> <tr> <td><code>ara::core::Result&lt; void &gt;</code></td> <td>–</td> </tr> </table>	<code>ara::core::Result&lt; void &gt;</code>	–
<code>ara::core::Result&lt; void &gt;</code>	–		





<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Register a notifier function which is called if the indicator is updated. A consecutive call of this method will overwrite the previous registered notifier.

]([RS\\_AP\\_00139](#), [RS\\_Diag\\_04204](#))

### 8.3.1.20 ServiceValidation class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticServiceValidationInterface](#).

Depending on attribute [serviceRequestCallbackType](#), the [ara::diag::ServiceValidation](#) is either called in the context of the [NRC](#) sequence in chapter 8.7 'Server response implementation rules' from ISO 14229-1[1] as a Manufacturer or a Supplier Check.

#### [SWS\_DM\_00771]{DRAFT} Definition of API class [ara::diag::ServiceValidation](#) [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/service_validation.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	ServiceValidation
<b>Syntax:</b>	<code>class ServiceValidation {...};</code>
<b>Description:</b>	<a href="#">DiagnosticServiceValidationInterface</a> .

]([RS\\_Diag\\_04199](#))

### 8.3.1.20.1 diag::ConfirmationStatusType

#### [SWS\_DM\_00770]{DRAFT} Definition of API enum [ara::diag::ConfirmationStatusType](#) [

<b>Kind:</b>	enumeration						
<b>Header file:</b>	#include "ara/diag/service_validation.h"						
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"						
<b>Scope:</b>	namespace ara::diag						
<b>Symbol:</b>	ConfirmationStatusType						
<b>Underlying type:</b>	std::uint8_t						
<b>Syntax:</b>	<code>enum class ConfirmationStatusType : std::uint8_t {...};</code>						
<b>Values:</b>	<table border="1"> <tr> <td>kResPosOk= 0x00</td> <td>Positive response has been sent out successfully.</td> </tr> <tr> <td>kResPosNotOk= 0x01</td> <td>Positive response has not been sent out successfully.</td> </tr> <tr> <td>kResNegOk= 0x02</td> <td>Negative response has been sent out successfull.</td> </tr> </table>	kResPosOk= 0x00	Positive response has been sent out successfully.	kResPosNotOk= 0x01	Positive response has not been sent out successfully.	kResNegOk= 0x02	Negative response has been sent out successfull.
kResPosOk= 0x00	Positive response has been sent out successfully.						
kResPosNotOk= 0x01	Positive response has not been sent out successfully.						
kResNegOk= 0x02	Negative response has been sent out successfull.						





	kResNegNotOk= 0x03	Negative response has not been sent out successfully.
	kResPosSuppressed= 0x04	Positive answer suppressed.
	kResNegSuppressed= 0x05	Negative answer suppressed.
	kCanceled= 0x06	Processing is canceled.
	kNoProcessingNo Response= 0x07	Processing rejected in Validation.
<b>Description:</b>	Represents the status of the service processing.	

]([RS\\_Diag\\_04199](#))

### 8.3.1.20.2 diag::ServiceValidation::ServiceValidation function

[SWS\_DM\_00772]{DRAFT} **Definition of API function ara::diag::ServiceValidation::ServiceValidation** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ServiceValidation</a>	
<b>Symbol:</b>	ServiceValidation(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit ServiceValidation (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticService ValidationInterface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of ServiceValidation.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04199](#))

### 8.3.1.20.3 diag::ServiceValidation::ServiceValidation(ServiceValidation&&) move constructor

[SWS\_DM\_01690]{DRAFT} **Definition of API function ara::diag::ServiceValidation::ServiceValidation** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ServiceValidation</a>	
<b>Symbol:</b>	ServiceValidation(ServiceValidation &&)	
<b>Syntax:</b>	ServiceValidation (ServiceValidation &&)=delete;	
<b>Description:</b>	Move constructor of ServiceValidation.	

]([RS\\_AP\\_00147](#))

### 8.3.1.20.4 diag::ServiceValidation::ServiceValidation(ServiceValidation&) copy constructor

[SWS\_DM\_01689][DRAFT] Definition of API function ara::diag::ServiceValidation::ServiceValidation [

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

](RS\_AP\_00147)

### 8.3.1.20.5 diag::ServiceValidation::ServiceValidation& operator=(ServiceValidation&&) move operator

[SWS\_DM\_01688][DRAFT] Definition of API function ara::diag::ServiceValidation::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/service_validation.h"
<b>Scope:</b>	class ara::diag::ServiceValidation
<b>Symbol:</b>	operator=(ServiceValidation &&)
<b>Syntax:</b>	ServiceValidation & operator= (ServiceValidation &&)=delete;
<b>Description:</b>	Move assignment operator of ServiceValidation.

](RS\_AP\_00147)

### 8.3.1.20.6 diag::ServiceValidation::ServiceValidation& operator=(ServiceValidation&) copy operator

[SWS\_DM\_01687][DRAFT] Definition of API function ara::diag::ServiceValidation::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/service_validation.h"
<b>Scope:</b>	class ara::diag::ServiceValidation
<b>Symbol:</b>	operator=(ServiceValidation &)
<b>Syntax:</b>	ServiceValidation & operator= (ServiceValidation &)=delete;
<b>Description:</b>	ServiceValidation shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.20.7 diag::ServiceValidation::~~ServiceValidation function

[SWS\_DM\_00773]{DRAFT} Definition of API function ara::diag::ServiceValidation::~~ServiceValidation [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/service_validation.h"
<b>Scope:</b>	<a href="#">class ara::diag::ServiceValidation</a>
<b>Symbol:</b>	~ServiceValidation()
<b>Syntax:</b>	virtual ~ServiceValidation () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of ServiceValidation.

] ([RS\\_AP\\_00134](#), [RS\\_Diag\\_04199](#))

### 8.3.1.20.8 diag::ServiceValidation::Validate function

[SWS\_DM\_00774]{DRAFT} Definition of API function ara::diag::ServiceValidation::Validate [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ServiceValidation</a>	
<b>Symbol:</b>	Validate(ara::core::Span< const std::uint8_t > requestData, const MetaInfo &metaInfo)	
<b>Syntax:</b>	virtual ara::core::Future< void > Validate (ara::core::Span< const std::uint8_t > requestData, const <a href="#">MetaInfo</a> &metaInfo);	
<b>Parameters (in):</b>	requestData	Diagnostic request data (including SID).
	metaInfo	MetaInfo of the request.
<b>Return value:</b>	ara::core::Future< void >	Returns nothing or an error
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.







DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.



△

	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoProcessingNo Response	0xFF, /// Deviating from ISO - no further service processing and no response (silently ignore request message).
<b>Description:</b>	Called for any request message.	

|(RS\_AP\_00138, RS\_Diag\_04170, RS\_Diag\_04199, RS\_AP\_00119)

### 8.3.1.20.9 diag::ServiceValidation::Confirmation function

[SWS\_DM\_00775]{DRAFT} **Definition of API function ara::diag::ServiceValidation::Confirmation** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	class ara::diag::ServiceValidation	
<b>Symbol:</b>	Confirmation(ConfirmationStatusType status, const MetalInfo &metalInfo)	
<b>Syntax:</b>	void Confirmation (ConfirmationStatusType status, const MetaInfo &meta Info);	
<b>Parameters (in):</b>	status	status/outcome of the service processing.
	metalInfo	MetalInfo of the request.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This method is called, when a diagnostic request has been finished, to notify about the outcome.	

|(RS\_AP\_00138, RS\_Diag\_04170, RS\_Diag\_04199)

### 8.3.1.20.10 diag::ServiceValidation::Offer function

[SWS\_DM\_00776]{DRAFT} Definition of API function ara::diag::ServiceValidation::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	class ara::diag::ServiceValidation	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	Returns nothing or an error
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04199](#))

### 8.3.1.20.11 diag::ServiceValidation::StopOffer function

[SWS\_DM\_00777]{DRAFT} Definition of API function ara::diag::ServiceValidation::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/service_validation.h"	
<b>Scope:</b>	class ara::diag::ServiceValidation	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	void StopOffer ();	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04199](#))

### 8.3.1.21 SecurityAccess class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticSecurityLevelInterface](#).

[SWS\_DM\_00761]{DRAFT} Definition of API class ara::diag::SecurityAccess [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	





<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SecurityAccess
<b>Syntax:</b>	<code>class SecurityAccess {...};</code>
<b>Description:</b>	DiagnosticSecurityAccessInterface.

|(RS\_Diag\_04005)

### 8.3.1.21.1 diag::KeyCompareResultType type

[SWS\_DM\_00760][DRAFT] Definition of API enum ara::diag::KeyCompareResultType [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	KeyCompareResultType	
<b>Underlying type:</b>	-	
<b>Syntax:</b>	enum class KeyCompareResultType {...};	
<b>Values:</b>	kKeyValid= 0x00	Key is valid.
	kKeyInvalid= 0x01	Key is invalid.
<b>Description:</b>	Represents the status of the key compare.	

|(RS\_Diag\_04005)

### 8.3.1.21.2 diag::SecurityAccess::SecurityAccess function

[SWS\_DM\_00762][DRAFT] Definition of API function ara::diag::SecurityAccess::SecurityAccess [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SecurityAccess</a>	
<b>Symbol:</b>	SecurityAccess(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit SecurityAccess (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticSecurity AccessInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of SecurityAccess.	

|(RS\_AP\_00137, RS\_Diag\_04005)

### 8.3.1.21.3 diag::SecurityAccess::SecurityAccess(SecurityAccess&&) move constructor

[SWS\_DM\_01618]{DRAFT} Definition of API function ara::diag::SecurityAccess::SecurityAccess [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/security_access.h"
<b>Scope:</b>	class ara::diag::SecurityAccess
<b>Symbol:</b>	SecurityAccess(SecurityAccess &&)
<b>Syntax:</b>	SecurityAccess (SecurityAccess &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of SecurityAccess.

](RS\_AP\_00147)

### 8.3.1.21.4 diag::SecurityAccess::SecurityAccess(SecurityAccess&) copy constructor

[SWS\_DM\_01617]{DRAFT} Definition of API function ara::diag::SecurityAccess::SecurityAccess [

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

](RS\_AP\_00147)

### 8.3.1.21.5 diag::SecurityAccess::SecurityAccess& operator=(SecurityAccess&&) move operator

[SWS\_DM\_01616]{DRAFT} Definition of API function ara::diag::SecurityAccess::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/security_access.h"
<b>Scope:</b>	class ara::diag::SecurityAccess
<b>Symbol:</b>	operator=(SecurityAccess &&)
<b>Syntax:</b>	SecurityAccess & operator= (SecurityAccess &&)=delete;
<b>Description:</b>	Move assignment operator of SecurityAccess.

](RS\_AP\_00147)

### 8.3.1.21.6 diag::SecurityAccess::SecurityAccess& operator=(SecurityAccess&) copy operator opera-

[SWS\_DM\_01615]{DRAFT} Definition of API function ara::diag::SecurityAccess::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/security_access.h"
<b>Scope:</b>	class ara::diag::SecurityAccess
<b>Symbol:</b>	operator=(SecurityAccess &)
<b>Syntax:</b>	SecurityAccess & operator= (SecurityAccess &)=delete;
<b>Description:</b>	SecurityAccess shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.21.7 diag::SecurityAccess::~~SecurityAccess function

[SWS\_DM\_00763]{DRAFT} Definition of API function ara::diag::SecurityAccess::~~SecurityAccess [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/security_access.h"
<b>Scope:</b>	class ara::diag::SecurityAccess
<b>Symbol:</b>	~SecurityAccess()
<b>Syntax:</b>	virtual ~SecurityAccess () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SecurityAccess.

](RS\_AP\_00134, RS\_Diag\_04005)

### 8.3.1.21.8 diag::SecurityAccess::GetSeed function

[SWS\_DM\_00764]{DRAFT} Definition of API function ara::diag::SecurityAccess::GetSeed [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Scope:</b>	class ara::diag::SecurityAccess	
<b>Symbol:</b>	GetSeed(ara::core::Span< const std::uint8_t > securityAccessDataRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< ara::core::Span< std::uint8_t > > GetSeed (ara::core::Span< const std::uint8_t > securityAccessDataRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	securityAccessData Record	Security Access payload



△

	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< ara::core::Span< std::uint8_t > >	provided seed
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
<b>Description:</b>	Called for any request message.	

|(RS\_AP\_00138, RS\_Diag\_04005, RS\_Diag\_04170, RS\_AP\_00119)



### 8.3.1.21.9 diag::SecurityAccess::CompareKey function

[SWS\_DM\_00765]{DRAFT} Definition of API function ara::diag::SecurityAccess::CompareKey [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Scope:</b>	class ara::diag::SecurityAccess	
<b>Symbol:</b>	CompareKey(ara::core::Span< const std::uint8_t > key, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< KeyCompareResultType > CompareKey(ara::core::Span< const std::uint8_t > key, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	key	The key to be validated
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< KeyCompareResultType >	Result of the key validation.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.





<b>Description:</b>	This method is called, when a diagnostic request has been finished, to notify about the outcome.
---------------------	--

|(RS\_AP\_00138, RS\_Diag\_04005, RS\_Diag\_04170, RS\_AP\_00119)

### 8.3.1.21.10 diag::SecurityAccess::Offer function

[SWS\_DM\_00766]{DRAFT} Definition of API function ara::diag::SecurityAccess::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Scope:</b>	class ara::diag::SecurityAccess	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

|(RS\_AP\_00139, RS\_Diag\_04005, RS\_AP\_00119)

### 8.3.1.21.11 diag::SecurityAccess::StopOffer function

[SWS\_DM\_00767]{DRAFT} Definition of API function ara::diag::SecurityAccess::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/security_access.h"	
<b>Scope:</b>	class ara::diag::SecurityAccess	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	void StopOffer ();	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

|(RS\_Diag\_04005)

### 8.3.1.22 CommunicationControl class

[SWS\_DM\_00804]{DRAFT} Definition of API class `ara::diag::CommunicationControl` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>CommunicationControl</code>
<b>Syntax:</b>	<code>class CommunicationControl {...};</code>
<b>Description:</b>	CommunicationControl interface.

]([RS\\_Diag\\_04196](#))

#### 8.3.1.22.1 `diag::CommunicationControl::ComCtrlRequestParamsType` type

[SWS\_DM\_00805]{DRAFT} Definition of API class `ara::diag::CommunicationControl::ComCtrlRequestParamsType` [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::CommunicationControl</code></a>
<b>Symbol:</b>	<code>ComCtrlRequestParamsType</code>
<b>Syntax:</b>	<code>struct ComCtrlRequestParamsType {...};</code>
<b>Description:</b>	<code>ComCtrlRequestParamsType</code> is a structure, which holds all parameters of an UDS 0x28 <code>communicationControl</code> request.

]([RS\\_Diag\\_04196](#))

#### 8.3.1.22.2 `diag::CommunicationControl::ComCtrlRequestParamsType::controlType` type

[SWS\_DM\_01562]{DRAFT} Definition of API variable `ara::diag::CommunicationControl::ComCtrlRequestParamsType::controlType` [

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

]([RS\\_Diag\\_04196](#))

### 8.3.1.22.3 diag::CommunicationControl::ComCtrlRequestParamsType-communicationType type

[SWS\_DM\_01563]{DRAFT} Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::communicationType [

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

]([RS\\_Diag\\_04196](#))

### 8.3.1.22.4 diag::CommunicationControl::ComCtrlRequestParamsType-nodeIdentificationNumber type

[SWS\_DM\_01564]{DRAFT} Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::nodeIdentificationNumber [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	<a href="#">struct ara::diag::CommunicationControl::ComCtrlRequestParamsType</a>
<b>Symbol:</b>	nodeIdentificationNumber
<b>Type:</b>	std::uint16_t
<b>Syntax:</b>	std::uint16_t nodeIdentificationNumber;
<b>Description:</b>	Node identifier to which the request is addressed to.

]([RS\\_Diag\\_04196](#))

### 8.3.1.22.5 diag::CommunicationControl::CommunicationControl function

[SWS\_DM\_00806]{DRAFT} Definition of API function ara::diag::CommunicationControl::CommunicationControl [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	<a href="#">class ara::diag::CommunicationControl</a>
<b>Symbol:</b>	CommunicationControl(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)
<b>Syntax:</b>	explicit CommunicationControl (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);





<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticComControl Interface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Class for an CommunicationControl.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04196](#))

### 8.3.1.22.6 diag::CommunicationControl::CommunicationControl(CommunicationControl&& move constructor

[SWS\_DM\_01678]{DRAFT} Definition of API function ara::diag::CommunicationControl::CommunicationControl [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	<a href="#">class ara::diag::CommunicationControl</a>
<b>Symbol:</b>	CommunicationControl(CommunicationControl &&)
<b>Syntax:</b>	CommunicationControl (CommunicationControl &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of CommunicationControl.

]([RS\\_AP\\_00147](#))

### 8.3.1.22.7 diag::CommunicationControl::CommunicationControl(CommunicationControl&) copy constructor

[SWS\_DM\_01677]{DRAFT} Definition of API function ara::diag::CommunicationControl::CommunicationControl [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	<a href="#">class ara::diag::CommunicationControl</a>
<b>Symbol:</b>	CommunicationControl(CommunicationControl &)
<b>Syntax:</b>	CommunicationControl (CommunicationControl &)=delete;
<b>Description:</b>	CommunicationControl shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.1.22.8 diag::CommunicationControl::CommunicationControl& operator=(CommunicationControl&&) move operator

[SWS\_DM\_01676]{DRAFT} Definition of API function ara::diag::CommunicationControl::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	class ara::diag::CommunicationControl
<b>Symbol:</b>	operator=(CommunicationControl &&)
<b>Syntax:</b>	CommunicationControl & operator= (CommunicationControl &&)=delete;
<b>Description:</b>	Move assignment operator of CommunicationControl.

](RS\_AP\_00147)

### 8.3.1.22.9 diag::CommunicationControl::CommunicationControl& operator=(CommunicationControl&) copy operator

[SWS\_DM\_01675]{DRAFT} Definition of API function ara::diag::CommunicationControl::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	class ara::diag::CommunicationControl
<b>Symbol:</b>	operator=(CommunicationControl &)
<b>Syntax:</b>	CommunicationControl & operator= (CommunicationControl &)=delete;
<b>Description:</b>	CommunicationControl shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.22.10 diag::CommunicationControl::~~CommunicationControl function

[SWS\_DM\_00807]{DRAFT} Definition of API function ara::diag::CommunicationControl::~~CommunicationControl [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/communication_control.h"
<b>Scope:</b>	class ara::diag::CommunicationControl
<b>Symbol:</b>	~CommunicationControl()
<b>Syntax:</b>	virtual ~CommunicationControl () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class CommunicationControl.

](RS\_AP\_00134, RS\_Diag\_04196)

### 8.3.1.22.11 diag::CommunicationControl::CommCtrlRequest function

#### [SWS\_DM\_00808]{DRAFT} Definition of API function ara::diag::CommunicationControl::CommCtrlRequest

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/communication_control.h"	
<b>Scope:</b>	class ara::diag::CommunicationControl	
<b>Symbol:</b>	CommCtrlRequest(ComCtrlRequestParamsType controlType, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > CommCtrlRequest (ComCtrlRequestParamsType controlType, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	controlType	All UDS request parameters packed into a structure since it holds optional elements
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
<b>Description:</b>	Called for CommunicationControl (x028) with any subfunction as subfunction value is part of argument list. Typically provider of this interface is considered as part of the state management.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04170, RS\_Diag\_04196)

### 8.3.1.22.12 diag::CommunicationControl::Offer function

[SWS\_DM\_00809]{DRAFT} Definition of API function ara::diag::CommunicationControl::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/communication_control.h"	
<b>Scope:</b>	class ara::diag::CommunicationControl	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04196](#))

### 8.3.1.22.13 diag::CommunicationControl::StopOffer function

[SWS\_DM\_00810]{DRAFT} Definition of API function ara::diag::CommunicationControl::StopOffer [

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

]([RS\\_Diag\\_04196](#))

### 8.3.1.23 DownloadService class

This interface is newly introduced.

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [Diagnostic-DownloadInterface](#).



**[SWS\_DM\_00784]{DRAFT} Definition of API class ara::diag::DownloadService** [

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

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

**8.3.1.23.1 diag::DownloadService::OperationOutput type**

**[SWS\_DM\_00785]{DRAFT} Definition of API class ara::diag::DownloadService::OperationOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/download.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>
<b>Symbol:</b>	OperationOutput
<b>Syntax:</b>	<code>struct OperationOutput {...};</code>
<b>Description:</b>	Response data of positive response message.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

**[SWS\_DM\_00786]{DRAFT} Definition of API variable ara::diag::DownloadService::OperationOutput::responseData** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/download.h"
<b>Scope:</b>	<a href="#">struct ara::diag::DownloadService::OperationOutput</a>
<b>Symbol:</b>	responseData
<b>Type:</b>	<code>ara::core::Vector&lt; std::uint8_t &gt;</code>
<b>Syntax:</b>	<code>ara::core::Vector&lt;std::uint8_t&gt; responseData;</code>
<b>Description:</b>	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.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.2 diag::DownloadService::DownloadService function

[SWS\_DM\_00787]{DRAFT} Definition of API function ara::diag::DownloadService::DownloadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>	
<b>Symbol:</b>	DownloadService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit DownloadService (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Class for an DownloadService.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.3 diag::DownloadService::DownloadService(DownloadService&&) move constructor

[SWS\_DM\_01650]{DRAFT} Definition of API function ara::diag::DownloadService::DownloadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>	
<b>Symbol:</b>	DownloadService(DownloadService &&)	
<b>Syntax:</b>	DownloadService (DownloadService &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of DownloadService.	

]([RS\\_AP\\_00147](#))

### 8.3.1.23.4 diag::DownloadService::DownloadService(DownloadService&) copy constructor

[SWS\_DM\_01649]{DRAFT} Definition of API function ara::diag::DownloadService::DownloadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>	





<b>Symbol:</b>	DownloadService(DownloadService &)
<b>Syntax:</b>	DownloadService (DownloadService &)=delete;
<b>Description:</b>	DownloadService shall be a single not copy-able instance.

](RS\_AP\_00147)

### 8.3.1.23.5 diag::DownloadService::DownloadService& operator=(DownloadService&&) move operator opera-

[SWS\_DM\_01648]{DRAFT} Definition of API function ara::diag::DownloadService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/download.h"
<b>Scope:</b>	class ara::diag::DownloadService
<b>Symbol:</b>	operator=(DownloadService &&)
<b>Syntax:</b>	DownloadService & operator= (DownloadService &&)=delete;
<b>Description:</b>	Move assignment operator of DownloadService.

](RS\_AP\_00147)

### 8.3.1.23.6 diag::DownloadService::DownloadService& operator=(DownloadService&) copy operator opera-

[SWS\_DM\_01647]{DRAFT} Definition of API function ara::diag::DownloadService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/download.h"
<b>Scope:</b>	class ara::diag::DownloadService
<b>Symbol:</b>	operator=(DownloadService &)
<b>Syntax:</b>	DownloadService & operator= (DownloadService &)=delete;
<b>Description:</b>	DownloadService shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.23.7 diag::DownloadService::~~DownloadService function

[SWS\_DM\_00788]{DRAFT} Definition of API function ara::diag::DownloadService::~~DownloadService [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/download.h"
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>
<b>Symbol:</b>	~DownloadService()
<b>Syntax:</b>	virtual ~DownloadService () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class DownloadService.

] ([RS\\_AP\\_00134](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.8 diag::DownloadService::RequestDownload function

[SWS\_DM\_00789]{DRAFT} Definition of API function ara::diag::DownloadService::RequestDownload [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>	
<b>Symbol:</b>	RequestDownload(std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > RequestDownload (std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	dataFormatIdentifier	UDS dataFormat Identifier
	addressAndLengthFormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >	a Future, which either gets readied to void (for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RequestDownload.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00138](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04170](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.9 diag::DownloadService::DownloadData function

[SWS\_DM\_00790]{DRAFT} Definition of API function ara::diag::DownloadService::DownloadData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DownloadService</a>	
<b>Symbol:</b>	DownloadData(ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > DownloadData (ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	transferRequestParameterRecord	data to be transferred (copied/downloaded to the ECU/server).
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< <a href="#">OperationOutput</a> >	a Future, which either gets readied to <a href="#">OperationOutput</a> (transferResponseParameterRecord for a positive response message) or readied with <a href="#">ErrorCode</a> from <a href="#">DiagUdsNrcErrc</a> (for an negative response message). Data in <a href="#">OperationOutput.responseData</a> will be placed after <a href="#">blockSequenceCounter</a> as <a href="#">transferResponseParameterRecord</a> in the positive response.
<b>Exception Safety:</b>	not exception safe	





<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for TransferData following a previous RequestDownload.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04033, RS\_Diag\_04170, RS\_Diag\_04196)

### 8.3.1.23.10 diag::DownloadService::RequestDownloadExit function

[SWS\_DM\_00791]{DRAFT} Definition of API function ara::diag::DownloadService::RequestDownloadExit [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	class ara::diag::DownloadService	
<b>Symbol:</b>	RequestDownloadExit(ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > RequestDownloadExit(ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	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.
	metaInfo	contains additional meta information





	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	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 a negative response message) Data in OperationOutput.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	Any	applicable NegativeResponseValue according to DiagUdsNrcErrc
<b>Description:</b>	Called for RequestTransferExit.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00138](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04170](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.11 diag::DownloadService::Offer function

[SWS\_DM\_00792]{DRAFT} Definition of API function ara::diag::DownloadService::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	class ara::diag::DownloadService	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.23.12 diag::DownloadService::StopOffer function

[SWS\_DM\_00793]{DRAFT} Definition of API function ara::diag::DownloadService::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/download.h"	
<b>Scope:</b>	class ara::diag::DownloadService	





<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	<code>void StopOffer ();</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.24 UploadService class

This interface is newly introduced.

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticUploadInterface](#).

#### [SWS\_DM\_00794]{DRAFT} Definition of API class `ara::diag::UploadService` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/upload.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>UploadService</code>
<b>Syntax:</b>	<code>class UploadService {...};</code>
<b>Description:</b>	Upload service interface.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

#### 8.3.1.24.1 `diag::UploadService::OperationOutput` type

#### [SWS\_DM\_00795]{DRAFT} Definition of API class `ara::diag::UploadService::OperationOutput` [

<b>Kind:</b>	struct
<b>Header file:</b>	<code>#include "ara/diag/upload.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::UploadService</code></a>
<b>Symbol:</b>	<code>OperationOutput</code>
<b>Syntax:</b>	<code>struct OperationOutput {...};</code>
<b>Description:</b>	Response data of positive response message.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))



### 8.3.1.24.2 diag::UploadService::UploadService function

[SWS\_DM\_00797]{DRAFT} Definition of API function ara::diag::UploadService::UploadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>	
<b>Symbol:</b>	UploadService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit UploadService (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Class for an UploadService.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.24.3 diag::UploadService::UploadService(UploadService&&) move constructor

[SWS\_DM\_01674]{DRAFT} Definition of API function ara::diag::UploadService::UploadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>	
<b>Symbol:</b>	UploadService(UploadService &&)	
<b>Syntax:</b>	UploadService (UploadService &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of UploadService.	

]([RS\\_AP\\_00147](#))

### 8.3.1.24.4 diag::UploadService::UploadService(UploadService&) copy constructor

[SWS\_DM\_01673]{DRAFT} Definition of API function ara::diag::UploadService::UploadService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>	



△

<b>Symbol:</b>	UploadService(UploadService &)
<b>Syntax:</b>	UploadService (UploadService &)=delete;
<b>Description:</b>	UploadService shall be a single not copy-able instance.

](RS\_AP\_00147)

### 8.3.1.24.5 diag::UploadService::UploadService& operator=(UploadService&&) move operator

[SWS\_DM\_01672]{DRAFT} Definition of API function ara::diag::UploadService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/upload.h"
<b>Scope:</b>	class ara::diag::UploadService
<b>Symbol:</b>	operator=(UploadService &&)
<b>Syntax:</b>	UploadService & operator= (UploadService &&)=delete;
<b>Description:</b>	Move assignment operator of UploadService.

](RS\_AP\_00147)

### 8.3.1.24.6 diag::UploadService::UploadService& operator=(UploadService&) copy operator

[SWS\_DM\_01671]{DRAFT} Definition of API function ara::diag::UploadService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/upload.h"
<b>Scope:</b>	class ara::diag::UploadService
<b>Symbol:</b>	operator=(UploadService &)
<b>Syntax:</b>	UploadService & operator= (UploadService &)=delete;
<b>Description:</b>	UploadService shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.24.7 diag::UploadService::~~UploadService function

[SWS\_DM\_00798]{DRAFT} Definition of API function ara::diag::UploadService::~~UploadService [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/upload.h"
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>
<b>Symbol:</b>	~UploadService()
<b>Syntax:</b>	virtual ~UploadService () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class UploadService.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.24.8 diag::UploadService::RequestUpload function

[SWS\_DM\_00799]{DRAFT} Definition of API function ara::diag::UploadService::RequestUpload [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>	
<b>Symbol:</b>	RequestUpload(std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > RequestUpload (std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< const std::uint8_t > memoryAddressAndSize, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	dataFormatIdentifier	UDS dataFormat Identifier
	addressAndLengthFormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RequestUpload.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00138](#), [RS\\_Diag\\_04033](#), [RS\\_Diag\\_04170](#), [RS\\_Diag\\_04196](#))

### 8.3.1.24.9 diag::UploadService::UploadData function

[SWS\_DM\_00800]{DRAFT} **Definition of API function ara::diag::UploadService::UploadData** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>	
<b>Symbol:</b>	UploadData(std::size_t numBytesToReturn, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > UploadData (std::size_t numBytesToReturn, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	numBytesToReturn	number of bytes DM accepts (due to its internal buffer) for this chunk.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< <a href="#">OperationOutput</a> > a Future, which either gets readied to <a href="#">OperationOutput</a> (transfer ResponseParameterRecord for a positive response message) or readied with <a href="#">ErrorCode</a> from <a href="#">DiagUdsNrcErrc</a> (for an negative response message). Data in <a href="#">OperationOutput.responseData</a> will be placed after <a href="#">blockSequenceCounter</a> as <a href="#">transferResponseParameterRecord</a> in the positive response.	
<b>Exception Safety:</b>	not exception safe	





<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for TransferData following a previous RequestUpload.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04033, RS\_Diag\_04170, RS\_Diag\_04196)

### 8.3.1.24.10 diag::UploadService::RequestUploadExit function

[SWS\_DM\_00801]{DRAFT} Definition of API function ara::diag::UploadService::RequestUploadExit [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	class ara::diag::UploadService	
<b>Symbol:</b>	RequestUploadExit(ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">OperationOutput</a> > RequestUploadExit(ara::core::Span< const std::uint8_t > transferRequestParameterRecord, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	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.
	metaInfo	contains additional meta information





	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	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 a negative response message) Data in OperationOutput.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RequestTransferExit.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04033, RS\_Diag\_04170, RS\_Diag\_04196)

### 8.3.1.24.11 diag::UploadService::Offer function

[SWS\_DM\_00802]{DRAFT} Definition of API function ara::diag::UploadService::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/upload.h"	
<b>Scope:</b>	class ara::diag::UploadService	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
	<b>Description:</b> This Offer will enable the DM to forward request messages to this handler.	

|(RS\_AP\_00119, RS\_AP\_00139, RS\_Diag\_04033, RS\_Diag\_04196)

### 8.3.1.24.12 diag::UploadService::StopOffer function

[SWS\_DM\_00803]{DRAFT} **Definition of API function ara::diag::UploadService::StopOffer** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/upload.h"
<b>Scope:</b>	<a href="#">class ara::diag::UploadService</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.24.13 diag::UploadService::OperationOutput responseData variable

[SWS\_DM\_00796]{DRAFT} **Definition of API variable ara::diag::UploadService::OperationOutput::responseData** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/upload.h"
<b>Scope:</b>	<a href="#">struct ara::diag::UploadService::OperationOutput</a>
<b>Symbol:</b>	responseData
<b>Type:</b>	ara::core::Vector< std::uint8_t >
<b>Syntax:</b>	ara::core::Vector<std::uint8_t> responseData;
<b>Description:</b>	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.

]([RS\\_Diag\\_04033](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25 EcuResetRequest class

[SWS\_DM\_01009]{DRAFT} **Definition of API class ara::diag::EcuResetRequest** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	EcuResetRequest
<b>Syntax:</b>	class EcuResetRequest {...};
<b>Description:</b>	Service EcuReset Request interface.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.1 diag::EcuResetRequest::ResetRequestType type

[SWS\_DM\_01007]{DRAFT} Definition of API type ara::diag::ResetRequestType [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	ResetRequestType
<b>Syntax:</b>	using ResetRequestType = std::uint8_t;
<b>Description:</b>	The type of the requested reset.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.2 diag::EcuResetRequest::ResetRequestType kCustomReset

[SWS\_DM\_01286]{DRAFT} Definition of API variable ara::diag::kCustomReset [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kCustomReset
<b>Type:</b>	ResetRequestType
<b>Syntax:</b>	constexpr ResetRequestType kCustomReset = 3;
<b>Description:</b>	kCustomReset

]([RS\\_AP\\_00125](#))

### 8.3.1.25.3 diag::EcuResetRequest::ResetRequestType kHardReset

[SWS\_DM\_01284]{DRAFT} Definition of API variable ara::diag::kHardReset [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kHardReset
<b>Type:</b>	ResetRequestType
<b>Syntax:</b>	constexpr ResetRequestType kHardReset = 1;
<b>Description:</b>	HardReset.

]([RS\\_AP\\_00125](#))



### 8.3.1.25.4 diag::EcuResetRequest::ResetRequestType kKeyOffOnReset

[SWS\_DM\_01285]{DRAFT} Definition of API variable ara::diag::kKeyOffOnReset

[

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kKeyOffOnReset
<b>Type:</b>	ResetRequestType
<b>Syntax:</b>	constexpr ResetRequestType kKeyOffOnReset = 2;
<b>Description:</b>	KeyOffOnReset.

]([RS\\_AP\\_00125](#))

### 8.3.1.25.5 diag::EcuResetRequest::ResetRequestType kSoftReset

[SWS\_DM\_01283]{DRAFT} Definition of API variable ara::diag::kSoftReset

[

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	kSoftReset
<b>Type:</b>	ResetRequestType
<b>Syntax:</b>	constexpr ResetRequestType kSoftReset = 0;
<b>Description:</b>	SoftReset.

]([RS\\_AP\\_00125](#))

### 8.3.1.25.6 diag::EcuResetRequest::StopOffer function

[SWS\_DM\_01017]{DRAFT} Definition of API function ara::diag::EcuResetRequest::StopOffer

[

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.7 diag::EcuResetRequest::Offer function

[SWS\_DM\_01016]{DRAFT} Definition of API function ara::diag::EcuResetRequest::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	class ara::diag::EcuResetRequest	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00139](#), [RS\\_AP\\_00119](#))

### 8.3.1.25.8 diag::EcuResetRequest::RequestReset function

[SWS\_DM\_01013]{DRAFT} Definition of API function ara::diag::EcuResetRequest::RequestReset [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	class ara::diag::EcuResetRequest	
<b>Symbol:</b>	RequestReset(ResetRequestType resetType, ara::core::Optional< std::uint8_t > id, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > RequestReset (ResetRequestType resetType, ara::core::Optional< std::uint8_t > id, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	resetType	Type of the requested reset.
	id	id of the custom reset type. Will only be evaluated when resetType is "custom"
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResourceTemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
<b>Description:</b>	Called for any EcuReset subFunction, except En-/DisableRapidShutdown. StateManagement needs to evaluate carefully if the request to restart parts or the whole machine. Once the request to reset is accepted, the StateManagement has to rely on this decision for the ExecuteReset() trigger.	

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.9 diag::EcuResetRequest::ExecuteReset function

[SWS\_DM\_01014]{DRAFT} Definition of API function ara::diag::EcuResetRequest::ExecuteReset [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>
<b>Symbol:</b>	ExecuteReset(const MetaInfo &metaInfo)
<b>Syntax:</b>	virtual ara::core::Future< void > ExecuteReset (const <a href="#">MetaInfo</a> &metaInfo)=0;





<b>DIRECTION NOT DEFINED</b>	metaInfo	-
<b>Return value:</b>	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
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
<b>Description:</b>	StateManagement has to execute the requested reset. Called after DM sent the response message to tester.	

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.10 diag::EcuResetRequest::EnableRapidShutdown function

[SWS\_DM\_01012]{DRAFT} Definition of API function ara::diag::EcuResetRequest::EnableRapidShutdown [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>	
<b>Symbol:</b>	EnableRapidShutdown(bool enable, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > EnableRapidShutdown (bool enable, const <a href="#">MetaInfo</a> &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	enable	when enable is set to true the rapid shutdown will be enabled, setting enable to false will disable rapid shutdown
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.







	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResource TemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
<b>Description:</b>	interface for subFunction En-/DisableRapidShutdown	

|(RS\_Diag\_04169, RS\_Diag\_04196)

### 8.3.1.25.11 diag::EcuResetRequest::~~EcuResetRequest destructor

[SWS\_DM\_01011]{DRAFT} Definition of API function ara::diag::EcuResetRequest::~~EcuResetRequest [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"
<b>Scope:</b>	class ara::diag::EcuResetRequest
<b>Symbol:</b>	~EcuResetRequest()
<b>Syntax:</b>	virtual ~EcuResetRequest () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of EcuResetRequest.

|(RS\_Diag\_04169, RS\_Diag\_04196)

### 8.3.1.25.12 diag::EcuResetRequest::EcuResetRequest constructor

[SWS\_DM\_01010]{DRAFT} Definition of API function ara::diag::EcuResetRequest::EcuResetRequest [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>	
<b>Symbol:</b>	EcuResetRequest(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit EcuResetRequest (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of EcuResetRequest.	

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04196](#))

### 8.3.1.25.13 diag::EcuResetRequest::EcuResetRequest(EcuResetRequest&&) move constructor

[SWS\_DM\_01622]{DRAFT} Definition of API function ara::diag::EcuResetRequest::EcuResetRequest [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>	
<b>Symbol:</b>	EcuResetRequest(EcuResetRequest &&)	
<b>Syntax:</b>	EcuResetRequest (EcuResetRequest &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of EcuResetRequest.	

]([RS\\_AP\\_00147](#))

### 8.3.1.25.14 diag::EcuResetRequest::EcuResetRequest(EcuResetRequest&) copy constructor

[SWS\_DM\_01621]{DRAFT} Definition of API function ara::diag::EcuResetRequest::EcuResetRequest [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/ecu_reset_request.h"	
<b>Scope:</b>	<a href="#">class ara::diag::EcuResetRequest</a>	
<b>Symbol:</b>	EcuResetRequest(EcuResetRequest &)	



△

<b>Syntax:</b>	<code>EcuResetRequest (EcuResetRequest &amp;)=delete;</code>
<b>Description:</b>	EcuResetRequest shall be a single not copy-able instance.

](RS\_AP\_00147)

### 8.3.1.25.15 diag::EcuResetRequest::EcuResetRequest& operator=(EcuResetRequest&&) move operator

[SWS\_DM\_01620]{DRAFT} Definition of API function `ara::diag::EcuResetRequest::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/ecu_reset_request.h"</code>
<b>Scope:</b>	<code>class ara::diag::EcuResetRequest</code>
<b>Symbol:</b>	<code>operator=(EcuResetRequest &amp;&amp;)</code>
<b>Syntax:</b>	<code>EcuResetRequest &amp; operator= (EcuResetRequest &amp;&amp;)=delete;</code>
<b>Description:</b>	Move assignment operator of EcuResetRequest.

](RS\_AP\_00147)

### 8.3.1.25.16 diag::EcuResetRequest::EcuResetRequest& operator=(EcuResetRequest&) copy operator

[SWS\_DM\_01619]{DRAFT} Definition of API function `ara::diag::EcuResetRequest::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/ecu_reset_request.h"</code>
<b>Scope:</b>	<code>class ara::diag::EcuResetRequest</code>
<b>Symbol:</b>	<code>operator=(EcuResetRequest &amp;)</code>
<b>Syntax:</b>	<code>EcuResetRequest &amp; operator= (EcuResetRequest &amp;)=delete;</code>
<b>Description:</b>	EcuResetRequest shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.1.26 Authentication class

#### [SWS\_DM\_01123]{DRAFT} Definition of API class `ara::diag::Authentication` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	Authentication
<b>Syntax:</b>	<code>class Authentication {...};</code>
<b>Description:</b>	Class to implement the Service Authentication interfaces to application.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.26.1 `diag::Authentication::Authentication` constructor

#### [SWS\_DM\_01124]{DRAFT} Definition of API function `ara::diag::Authentication::Authentication` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/authentication.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::Authentication</code></a>	
<b>Symbol:</b>	<code>Authentication(const ara::core::InstanceSpecifier &amp;specifier, ReentrancyType reentrancyType)</code>	
<b>Syntax:</b>	<code>explicit Authentication (const ara::core::InstanceSpecifier &amp;specifier, <a href="#">ReentrancyType</a> reentrancyType);</code>	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to a PortPrototype of a DiagnosticAuthentication service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of Authentication.	

]([RS\\_Diag\\_04251](#))

#### 8.3.1.26.2 `diag::Authentication::Authentication(Authentication&&)` move constructor

#### [SWS\_DM\_01610]{DRAFT} Definition of API function `ara::diag::Authentication::Authentication` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::Authentication</code></a>
<b>Symbol:</b>	<code>Authentication(Authentication &amp;&amp;)</code>
<b>Syntax:</b>	<code>Authentication (Authentication &amp;&amp;) noexcept=delete;</code>



△

<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of Authentication.

](RS\_AP\_00147)

### 8.3.1.26.3 diag::Authentication::Authentication(Authentication&) copy constructor

[SWS\_DM\_01609]{DRAFT} Definition of API function ara::diag::Authentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::Authentication</a>
<b>Symbol:</b>	Authentication(Authentication &)
<b>Syntax:</b>	Authentication (Authentication &)=delete;
<b>Description:</b>	Authentication shall be a single not copy-able instance.

](RS\_AP\_00147)

### 8.3.1.26.4 diag::Authentication::Authentication& operator=(Authentication&&) move operator

[SWS\_DM\_01608]{DRAFT} Definition of API function ara::diag::Authentication::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::Authentication</a>
<b>Symbol:</b>	operator=(Authentication &&)
<b>Syntax:</b>	Authentication & operator= (Authentication &&)=delete;
<b>Description:</b>	Move assignment operator of Authentication.

](RS\_AP\_00147)

### 8.3.1.26.5 diag::Authentication::Authentication& operator=(Authentication&) copy operator

[SWS\_DM\_01607]{DRAFT} Definition of API function  
ara::diag::Authentication::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	class ara::diag::Authentication
<b>Symbol:</b>	operator=(Authentication &)
<b>Syntax:</b>	Authentication & operator= (Authentication &)=delete;
<b>Description:</b>	Authentication shall be a single not assignable instance.

] ([RS\\_AP\\_00147](#))

### 8.3.1.26.6 diag::Authentication::~~Authentication destructor

[SWS\_DM\_01125]{DRAFT} Definition of API function  
ara::diag::Authentication::~~Authentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	class ara::diag::Authentication
<b>Symbol:</b>	~Authentication()
<b>Syntax:</b>	virtual ~Authentication () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of Authentication.

] ([RS\\_Diag\\_04251](#))

### 8.3.1.26.7 diag::Authentication::VerifyCertificateUnidirectional function

[SWS\_DM\_01126]{DRAFT} Definition of API function  
ara::diag::Authentication::VerifyCertificateUnidirectional [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/authentication.h"
<b>Scope:</b>	class ara::diag::Authentication
<b>Symbol:</b>	VerifyCertificateUnidirectional(ara::core::Byte communicationConfiguration, ara::core::Span< ara::core::Byte > clientCertificate, ara::core::Span< ara::core::Byte > clientChallenge, const MetalInfo &metalInfo, CancellationHandler cancellationHandler)





<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; std::tuple&lt; ara::core::Span&lt; ara::core::Byte &gt;, ara::core::Span&lt; ara::core::Byte &gt; &gt; &gt; Verify CertificateUnidirectional (ara::core::Byte communicationConfiguration, ara::core::Span&lt; ara::core::Byte &gt; clientCertificate, ara::core::Span&lt; ara::core::Byte &gt; clientChallenge, const MetaInfo &amp;metaInfo, CancellationHandler cancellationHandler)=0;</pre>	
<b>Parameters (in):</b>	communication Configuration	As defined in ISO14229-1:2020, this parameter provides information about how to proceed with security in further diagnostic communication after the Authentication.
	clientCertificate	The certificate that is received from the tester during Unidirectional Authentication.
	clientChallenge	As defined in ISO14229-1:2020, this parameter provides the challenge received from the tester during Unidirectional Authentication. This parameter has a dependency on the CommunicationConfiguration used
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	<pre>ara::core::Future&lt; std::tuple&lt; ara::core::Span&lt; ara::core::Byte &gt;, ara::core::Span&lt; ara::core::Byte &gt; &gt; &gt;</pre>	Challenge created by the application, Ephemeral Public Key of Server
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid TimePeriod</pre>	Date and time of the server does not match the validity period of the Certificate.
	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Signature</pre>	Signature of the Certificate could not be verified.
	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid ChainOfTrust</pre>	Certificate could not be verified against stored information about the issuing authority.
	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Type</pre>	Certificate does not match the current requested use case.
	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Format</pre>	Certificate could not be evaluated because the format requirement has not been met.
	<pre>DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Content</pre>	Certificate could not be verified because the content does not match.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Scope	The scope of the Certificate does not match the contents of the server.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Cert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kChallenge CalculationFailed	The challenge could not be calculated on the server side.
<b>Description:</b>	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.	

]([RS\\_Diag\\_04251](#), [RS\\_AP\\_00128](#))

### 8.3.1.26.8 diag::Authentication::VerifyCertificateBidirectional function

[SWS\_DM\_01127]{DRAFT} **Definition of API function**  
**ara::diag::Authentication::VerifyCertificateBidirectional** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/authentication.h"	
<b>Scope:</b>	class ara::diag::Authentication	
<b>Symbol:</b>	VerifyCertificateBidirectional(ara::core::Byte communicationConfiguration, ara::core::Span< ara::core::Byte > clientCertificate, ara::core::Span< ara::core::Byte > clientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< std::tuple< ara::core::Span< ara::core::Byte >, ara::core::Span< ara::core::Byte >, ara::core::Span< ara::core::Byte >, ara::core::Span< ara::core::Byte > > > VerifyCertificateBidirectional (ara::core::Byte communicationConfiguration, ara::core::Span< ara::core::Byte > clientCertificate, ara::core::Span< ara::core::Byte > clientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	communicationConfiguration	As defined in ISO14229-1:2020, this parameter provides information about how to proceed with security in further diagnostic communication after the Authentication.
	clientCertificate	The certificate that is received from the tester during Bidirectional Authentication.
	clientChallenge	As defined in ISO14229-1:2020, this parameter provides the challenge received from the tester during Bidirectional Authentication.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..







<b>Return value:</b>	ara::core::Future<std::tuple<ara::core::Span<ara::core::Byte >, ara::core::Span<ara::core::Byte >, ara::core::Span<ara::core::Byte >, ara::core::Span<ara::core::Byte >>>	Challenge created by the application, Certificate of the server, ProofOfOwnership calculated by the server, Ephemeral Public Key of server
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidTimePeriod	Date and time of the server does not match the validity period of the Certificate.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidSignature	Signature of the Certificate could not be verified.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidChainOfTrust	Certificate could not be verified against stored information about the issuing authority.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidType	Certificate does not match the current requested use case.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidFormat	Certificate could not be evaluated because the format requirement has not been met.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidContent	Certificate could not be verified because the content does not match.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidScope	The scope of the Certificate does not match the contents of the server.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidCert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kChallengeCalculationFailed	The challenge could not be calculated on the server side.
<b>Description:</b>	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.	

|(RS\_Diag\_04251, RS\_AP\_00128)

### 8.3.1.26.9 diag::Authentication::VerifyOwnership function

#### [SWS\_DM\_01128]{DRAFT} Definition of API function ara::diag::Authentication::VerifyOwnership [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/authentication.h"	
<b>Scope:</b>	class ara::diag::Authentication	
<b>Symbol:</b>	VerifyOwnership(ara::core::Span< ara::core::Byte > clientPOWN, ara::core::Span< ara::core::Byte > clientEphemeralPublicKey, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< ara::core::Span< ara::core::Byte > > VerifyOwnership (ara::core::Span< ara::core::Byte > clientPOWN, ara::core::Span< ara::core::Byte > clientEphemeralPublicKey, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	clientPOWN	The Proof Of Ownership provided by the Tester to the previously exchanged Server Challenge
	clientEphemeralPublicKey	As defined in ISO14229-1:2020, this is the Ephemeral public key generated by the client for Diffie-Hellman key agreement.
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< ara::core::Span< ara::core::Byte > >	Session Key Info or error
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidTimePeriod	Date and time of the server does not match the validity period of the Certificate.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidSignature	Signature of the Certificate could not be verified.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidChainOfTrust	Certificate could not be verified against stored information about the issuing authority.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidType	Certificate does not match the current requested use case.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificateVerificationFailedInvalidFormat	Certificate could not be evaluated because the format requirement has not been met.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Content	Certificate could not be verified because the content does not match.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Scope	The scope of the Certificate does not match the contents of the server.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kCertificate VerificationFailedInvalid Cert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kChallenge CalculationFailed	The challenge could not be calculated on the server side.
<b>Description:</b>	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.	

](RS\_Diag\_04251, RS\_AP\_00128)

### 8.3.1.26.10 diag::Authentication::TransmitCertificate function

[SWS\_DM\_01129]{DRAFT} Definition of API function  
 ara::diag::Authentication::TransmitCertificate [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/authentication.h"	
<b>Scope:</b>	class ara::diag::Authentication	
<b>Symbol:</b>	TransmitCertificate(ara::core::Span< ara::core::Byte > certificateData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > TransmitCertificate (ara::core::Span< ara::core::Byte > certificateData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	certificateData	Certificate to be verified, transmitted by the tester
	metaInfo	MetaInfo of the request.
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled)..
<b>Return value:</b>	ara::core::Future< void >	void
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kOwnership VerificationFailed	Delivered Ownership does not match the provided challenge or could not be verified with the own private key.





<b>Description:</b>	This function accepts the certificate received from the tester, verifies it, and takes necessary actions in the applications. Typical use cases include updates to the access-rights provided to a tester that is already authenticated.
---------------------	--

](RS\_Diag\_04251)

### 8.3.1.26.11 diag::Authentication::Offer function

[SWS\_DM\_01130]{DRAFT} Definition of API function  
ara::diag::Authentication::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/authentication.h"	
<b>Scope:</b>	class ara::diag::Authentication	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

](RS\_Diag\_04251)

### 8.3.1.26.12 diag::Authentication::StopOffer function

[SWS\_DM\_01131]{DRAFT} Definition of API function  
ara::diag::Authentication::StopOffer [

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

](RS\_Diag\_04251)

### 8.3.1.27 ClientAuthentication class

#### [SWS\_DM\_01132]{DRAFT} Definition of API class ara::diag::ClientAuthentication

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/client_authentication.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	ClientAuthentication
<b>Syntax:</b>	<code>class ClientAuthentication final {...};</code>
<b>Description:</b>	Interface for the application to inform the Diagnostic Server instance about the authentication states and the user roles that are currently authenticated.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.27.1 diag::ClientAuthentication::DiagnosticAuthState type

#### [SWS\_DM\_01133]{DRAFT} Definition of API enum ara::diag::ClientAuthentication::DiagnosticAuthState

<b>Kind:</b>	enumeration				
<b>Header file:</b>	#include "ara/diag/client_authentication.h"				
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"				
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>				
<b>Symbol:</b>	DiagnosticAuthState				
<b>Underlying type:</b>	std::uint8_t				
<b>Syntax:</b>	<code>enum class DiagnosticAuthState : std::uint8_t {...};</code>				
<b>Values:</b>	<table border="1"> <tr> <td>kDeAuthenticated= 0x00</td> <td>No Diagnostic Clients are currently authenticated.</td> </tr> <tr> <td>kAuthenticated= 0x01</td> <td>A Diagnostic Client is currently authenticated.</td> </tr> </table>	kDeAuthenticated= 0x00	No Diagnostic Clients are currently authenticated.	kAuthenticated= 0x01	A Diagnostic Client is currently authenticated.
kDeAuthenticated= 0x00	No Diagnostic Clients are currently authenticated.				
kAuthenticated= 0x01	A Diagnostic Client is currently authenticated.				
<b>Description:</b>	Possible values of the Authentication State of the client.				

]([RS\\_Diag\\_04251](#))

#### 8.3.1.27.2 diag::ClientAuthentication::DiagnosticAuthRole type

#### [SWS\_DM\_01134]{DRAFT} Definition of API type ara::diag::ClientAuthentication::DiagnosticAuthRole

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/client_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>
<b>Symbol:</b>	DiagnosticAuthRole
<b>Syntax:</b>	<code>using DiagnosticAuthRole = ara::core::String;</code>
<b>Description:</b>	The Supported values for the Diagnostic Authentication roles are specified in the Diagnostic Extract.

](RS\_Diag\_04251)

### 8.3.1.27.3 diag::ClientAuthentication::~~ClientAuthentication destructor

[SWS\_DM\_01136][DRAFT] Definition of API function ara::diag::ClientAuthentication::~~ClientAuthentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication.h"
<b>Scope:</b>	class ara::diag::ClientAuthentication
<b>Symbol:</b>	~ClientAuthentication()
<b>Syntax:</b>	~ClientAuthentication () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of ClientAuthentication.

](RS\_Diag\_04251)

### 8.3.1.27.4 diag::ClientAuthentication::ClientAuthentication copy constructor

[SWS\_DM\_01139][DRAFT] Definition of API function ara::diag::ClientAuthentication::ClientAuthentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication.h"
<b>Scope:</b>	class ara::diag::ClientAuthentication
<b>Symbol:</b>	ClientAuthentication(ClientAuthentication const &other)
<b>Syntax:</b>	ClientAuthentication (ClientAuthentication const &other)=delete;
<b>Description:</b>	Copy constructor of ClientAuthentication cannot be used.

](RS\_Diag\_04251)

### 8.3.1.27.5 diag::ClientAuthentication::ClientAuthentication move constructor

[SWS\_DM\_01137][DRAFT] Definition of API function ara::diag::ClientAuthentication::ClientAuthentication [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	class ara::diag::ClientAuthentication	
<b>Symbol:</b>	ClientAuthentication(ClientAuthentication &&other)	
<b>Syntax:</b>	ClientAuthentication (ClientAuthentication &&other) noexcept=default;	
<b>Parameters (in):</b>	other	Object to move-construct from



△

<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of ClientAuthentication.

 ]([RS\\_Diag\\_04251](#))

### 8.3.1.27.6 diag::ClientAuthentication::ClientAuthentication copy assignment operator

[SWS\_DM\_01140]{DRAFT} Definition of API function `ara::diag::ClientAuthentication::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>
<b>Symbol:</b>	<code>operator=(ClientAuthentication const &amp;other)</code>
<b>Syntax:</b>	<code>auto operator= (ClientAuthentication const &amp;other) -&gt; ClientAuthentication &amp;=delete;</code>
<b>Description:</b>	Copy assignment operator of ClientAuthentication cannot be used.

 ]([RS\\_Diag\\_04251](#))

### 8.3.1.27.7 diag::ClientAuthentication::ClientAuthentication move assignment operator

[SWS\_DM\_01138]{DRAFT} Definition of API function `ara::diag::ClientAuthentication::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>	
<b>Symbol:</b>	<code>operator=(ClientAuthentication &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>auto operator= (ClientAuthentication &amp;&amp;other) &amp; noexcept -&gt; ClientAuthentication &amp;=default;</code>	
<b>Parameters (in):</b>	other	Object to move-assign from.
<b>Return value:</b>	ClientAuthentication &=default	-
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assignment operator of ClientAuthentication.	

 ]([RS\\_Diag\\_04251](#))

### 8.3.1.27.8 diag::ClientAuthentication::OverrideDefaultRoles function

[SWS\_DM\_01141]{DRAFT} Definition of API function ara::diag::ClientAuthentication::OverrideDefaultRoles [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>	
<b>Symbol:</b>	OverrideDefaultRoles(ara::core::Vector< DiagnosticAuthRole > defaultRoles, std::chrono::milliseconds timeout)	
<b>Syntax:</b>	<pre>ara::core::Result&lt; ClientAuthenticationHandle &gt; OverrideDefaultRoles (ara::core::Vector&lt; DiagnosticAuthRole &gt; defaultRoles, std::chrono::milliseconds timeout);</pre>	
<b>Parameters (in):</b>	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
<b>Return value:</b>	ara::core::Result< Client AuthenticationHandle >	Operation result
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.27.9 diag::ClientAuthentication::Authenticate function

[SWS\_DM\_01142]{DRAFT} Definition of API function ara::diag::ClientAuthentication::Authenticate [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>	
<b>Symbol:</b>	Authenticate(ara::core::Vector< DiagnosticAuthRole > userRoles)	
<b>Syntax:</b>	<pre>ara::core::Result&lt; ClientAuthenticationHandle &gt; Authenticate (ara::core::Vector&lt; DiagnosticAuthRole &gt; userRoles);</pre>	
<b>Parameters (in):</b>	userRoles	The user roles to set on the diagnostic client
		A handler of the Authentication State, which can be used by the application to set or extend the DynamicAccessList
<b>Return value:</b>	ara::core::Result< Client AuthenticationHandle >	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04251](#))



### 8.3.1.27.10 diag::ClientAuthentication::GetState function

[SWS\_DM\_01143]{DRAFT} Definition of API function ara::diag::ClientAuthentication::GetState [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>	
<b>Symbol:</b>	GetState()	
<b>Syntax:</b>	ara::core::Result< <a href="#">DiagnosticAuthState</a> > GetState () const noexcept;	
<b>Return value:</b>	ara::core::Result< DiagnosticAuthState >	The Authentication State of the Diagnostic Client or error
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to query the current authentication state of the diagnostic client.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.27.11 diag::ClientAuthentication::SetNotifier function

[SWS\_DM\_01144]{DRAFT} Definition of API function ara::diag::ClientAuthentication::SetNotifier [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthentication</a>	
<b>Symbol:</b>	SetNotifier(std::function< void(DiagnosticAuthState)> notifier)	
<b>Syntax:</b>	ara::core::Result< void > SetNotifier (std::function< void( <a href="#">DiagnosticAuthState</a> )> notifier);	
<b>Parameters (in):</b>	notifier	The notifier to call on state transition
<b>Return value:</b>	ara::core::Result< void >	void
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This function is used by the application to set a Notifier Function that shall be called by the Diagnostic Server instance when an Authentication Status Change Occurs. This may be used, for. E.g, to notify the application when a transition to kDeAuthenticated State occurred due to an S3 timeout. A consecutive call of this method will overwrite the previous registered notifier.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.28 ClientAuthenticationHandle class

#### [SWS\_DM\_01145]{DRAFT} Definition of API class ara::diag::ClientAuthenticationHandle [

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

]([RS\\_Diag\\_04251](#))

#### 8.3.1.28.1 diag::ClientAuthenticationHandle::ClientAuthenticationHandle constructor

#### [SWS\_DM\_01146]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthenticationHandle</a>
<b>Symbol:</b>	ClientAuthenticationHandle()
<b>Syntax:</b>	<code>ClientAuthenticationHandle ();</code>
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Constructor of ClientAuthenticationHandle.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.28.2 diag::ClientAuthenticationHandle::~~ClientAuthenticationHandle destructor

#### [SWS\_DM\_01147]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::~~ClientAuthenticationHandle [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthenticationHandle</a>
<b>Symbol:</b>	~ClientAuthenticationHandle()
<b>Syntax:</b>	<code>~ClientAuthenticationHandle () noexcept=default;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of ClientAuthenticationHandle.

](RS\_Diag\_04251)

### 8.3.1.28.3 diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy constructor

[SWS\_DM\_01150]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	class ara::diag::ClientAuthenticationHandle
<b>Symbol:</b>	ClientAuthenticationHandle(ClientAuthenticationHandle const &other)
<b>Syntax:</b>	ClientAuthenticationHandle (ClientAuthenticationHandle const &other)=delete;
<b>Description:</b>	Copy constructor of ClientAuthenticationHandle cannot be used.

](RS\_Diag\_04251)

### 8.3.1.28.4 diag::ClientAuthenticationHandle::ClientAuthenticationHandle move constructor

[SWS\_DM\_01148]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::ClientAuthenticationHandle [

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

](RS\_Diag\_04251)

### 8.3.1.28.5 diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy assignment operator

[SWS\_DM\_01151]{DRAFT} Definition of API function `ara::diag::ClientAuthenticationHandle::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	<code>class ara::diag::ClientAuthenticationHandle</code>
<b>Symbol:</b>	<code>operator=(ClientAuthenticationHandle const &amp;other)</code>
<b>Syntax:</b>	<code>auto operator= (ClientAuthenticationHandle const &amp;other) -&gt; ClientAuthenticationHandle &amp;=delete;</code>
<b>Description:</b>	Copy assignment operator of CancellationHandler cannot be used.

]([RS\\_Diag\\_04251](#))

### 8.3.1.28.6 diag::ClientAuthenticationHandle::ClientAuthenticationHandle move assignment operator

[SWS\_DM\_01149]{DRAFT} Definition of API function `ara::diag::ClientAuthenticationHandle::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"	
<b>Scope:</b>	<code>class ara::diag::ClientAuthenticationHandle</code>	
<b>Symbol:</b>	<code>operator=(ClientAuthenticationHandle &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>auto operator= (ClientAuthenticationHandle &amp;&amp;other) &amp; noexcept -&gt; ClientAuthenticationHandle &amp; =default;</code>	
<b>Parameters (in):</b>	other	Object to move-assign from.
<b>Return value:</b>	ClientAuthenticationHandle & =default	-
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assignment operator of ClientAuthenticationHandle.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.28.7 diag::ClientAuthenticationHandle::Append function

[SWS\_DM\_01152]{DRAFT} Definition of API function `ara::diag::ClientAuthenticationHandle::Append` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	<code>class ara::diag::ClientAuthenticationHandle</code>
<b>Symbol:</b>	<code>Append(DiagnosticServiceDynamicAccessList dynamicAccessList)</code>





<b>Syntax:</b>	ara::core::Result< void > Append (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;	
<b>Parameters (in):</b>	dynamicAccessList	The DynamicAccessList to be appended in the client
<b>Return value:</b>	ara::core::Result< void >	void
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to append a DynamicAccessList to the already existing DynamicAccessList of a Diagnostic Conversation.	

|(RS\_Diag\_04251)

### 8.3.1.28.8 diag::ClientAuthenticationHandle::Set function

[SWS\_DM\_01153]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::Set [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"	
<b>Scope:</b>	class ara::diag::ClientAuthenticationHandle	
<b>Symbol:</b>	Set(DiagnosticServiceDynamicAccessList dynamicAccessList)	
<b>Syntax:</b>	ara::core::Result< void > Set (DiagnosticServiceDynamicAccessList dynamicAccessList) noexcept;	
<b>Parameters (in):</b>	dynamicAccessList	The new DynamicAccessList to be set in the client
<b>Return value:</b>	ara::core::Result< void >	void
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to set/replace a DynamicAccessList of a diagnostic conversation.	

|(RS\_Diag\_04251)

### 8.3.1.28.9 diag::ClientAuthenticationHandle::Revoke function

[SWS\_DM\_01154]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::Revoke [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"	
<b>Scope:</b>	class ara::diag::ClientAuthenticationHandle	
<b>Symbol:</b>	Revoke()	
<b>Syntax:</b>	ara::core::Result< void > Revoke () noexcept;	
<b>Return value:</b>	ara::core::Result< void >	void
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to de-authenticate a client, and also to clear the Dynamic AccessList and any overridden defaults.	

|(RS\_Diag\_04251)

### 8.3.1.28.10 diag::ClientAuthenticationHandle::Refresh function

[SWS\_DM\_01155]{DRAFT} Definition of API function ara::diag::ClientAuthenticationHandle::Refresh [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/client_authentication_handle.h"
<b>Scope:</b>	<a href="#">class ara::diag::ClientAuthenticationHandle</a>
<b>Symbol:</b>	Refresh()
<b>Syntax:</b>	ara::core::Result< void > Refresh () noexcept;
<b>Return value:</b>	ara::core::Result< void >   void
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	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.

]([RS\\_Diag\\_04251](#))

### 8.3.1.29 DiagnosticServiceDynamicAccessList class

[SWS\_DM\_01156]{DRAFT} Definition of API class ara::diag::DiagnosticServiceDynamicAccessList [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DiagnosticServiceDynamicAccessList
<b>Syntax:</b>	class DiagnosticServiceDynamicAccessList final {...};
<b>Description:</b>	Definition of the DiagnosticServiceDynamicAccessList class, which is used by the application to build a DynamicAccessList.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.29.1 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList constructor

[SWS\_DM\_01157]{DRAFT} Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
<b>Scope:</b>	<a href="#">class ara::diag::DiagnosticServiceDynamicAccessList</a>
<b>Symbol:</b>	DiagnosticServiceDynamicAccessList()
<b>Syntax:</b>	DiagnosticServiceDynamicAccessList ();





<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	Constructor of DiagnosticServiceDynamicAccessList.

](RS\_Diag\_04251)

### 8.3.1.29.2 diag::DiagnosticServiceDynamicAccessList::~~DiagnosticServiceDynamicAccessList destructor

[SWS\_DM\_01158]{DRAFT} Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::~~DiagnosticServiceDynamicAccessList [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
<b>Scope:</b>	class ara::diag::DiagnosticServiceDynamicAccessList
<b>Symbol:</b>	~DiagnosticServiceDynamicAccessList()
<b>Syntax:</b>	~DiagnosticServiceDynamicAccessList () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DiagnosticServiceDynamicAccessList.

](RS\_Diag\_04251)

### 8.3.1.29.3 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy constructor

[SWS\_DM\_01159]{DRAFT} Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
<b>Scope:</b>	class ara::diag::DiagnosticServiceDynamicAccessList
<b>Symbol:</b>	DiagnosticServiceDynamicAccessList(DiagnosticServiceDynamicAccessList const &other)
<b>Syntax:</b>	DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList const &other) noexcept;
<b>Parameters (in):</b>	other   Object to copy-construct from
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Copy constructor of DiagnosticServiceDynamicAccessList.

](RS\_Diag\_04251)

### 8.3.1.29.4 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move constructor

[SWS\_DM\_01160]{DRAFT} Definition of API function `ara::diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class <code>ara::diag::DiagnosticServiceDynamicAccessList</code>	
<b>Symbol:</b>	<code>DiagnosticServiceDynamicAccessList(DiagnosticServiceDynamicAccessList &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList &amp;&amp;other) noexcept;</code>	
<b>Parameters (in):</b>	other	Object to move-construct from
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of <code>DiagnosticServiceDynamicAccessList</code> .	

]([RS\\_Diag\\_04251](#))

### 8.3.1.29.5 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy assignment operator

[SWS\_DM\_01161]{DRAFT} Definition of API function `ara::diag::DiagnosticServiceDynamicAccessList::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class <code>ara::diag::DiagnosticServiceDynamicAccessList</code>	
<b>Symbol:</b>	<code>operator=(DiagnosticServiceDynamicAccessList const &amp;other)</code>	
<b>Syntax:</b>	<code>auto operator= (DiagnosticServiceDynamicAccessList const &amp;other) &amp; -&gt; DiagnosticServiceDynamicAccessList &amp;;</code>	
<b>Parameters (in):</b>	other	Object to copy-assign from.
<b>Return value:</b>	<code>DiagnosticServiceDynamicAccessList &amp;</code>	-
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Copy assignment operator of <code>DiagnosticServiceDynamicAccessList</code> .	

]([RS\\_Diag\\_04251](#))



### 8.3.1.29.6 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move assignment operator

[SWS\_DM\_01162]{DRAFT} Definition of API function `ara::diag::DiagnosticServiceDynamicAccessList::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class <code>ara::diag::DiagnosticServiceDynamicAccessList</code>	
<b>Symbol:</b>	<code>operator=(DiagnosticServiceDynamicAccessList &amp;&amp;other)</code>	
<b>Syntax:</b>	<pre>auto operator= (DiagnosticServiceDynamicAccessList &amp;&amp;other) &amp; -&gt; DiagnosticServiceDynamicAccessList &amp;;</pre>	
<b>Parameters (in):</b>	other	Object to move-assign from.
<b>Return value:</b>	DiagnosticServiceDynamicAccessList &	-
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Move assignment operator of DiagnosticServiceDynamicAccessList.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.29.7 diag::DiagnosticServiceDynamicAccessList::Reserve function

[SWS\_DM\_01163]{DRAFT} Definition of API function `ara::diag::DiagnosticServiceDynamicAccessList::Reserve` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class <code>ara::diag::DiagnosticServiceDynamicAccessList</code>	
<b>Symbol:</b>	<code>Reserve(std::size_t numberOfServiceHeads, std::size_t maxServiceHeadSize)</code>	
<b>Syntax:</b>	<pre>void Reserve (std::size_t numberOfServiceHeads, std::size_t maxServiceHeadSize) noexcept;</pre>	
<b>Parameters (in):</b>	numberOfServiceHeads	The number of diagnostic service patterns
	maxServiceHeadSize	The expected maximum number of diagnostic service bytes in a single pattern
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.29.8 diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function

[SWS\_DM\_01164]{DRAFT} Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class ara::diag::DiagnosticServiceDynamicAccessList	
<b>Symbol:</b>	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::Byte sid)	
<b>Syntax:</b>	auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte sid) noexcept -> DynamicAccessListDiagServiceBuilder;	
<b>Parameters (in):</b>	sid	The diagnostic service identifier
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder	An instance of a diagnostic service pattern builder
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the Application to construct a pattern for the DynamicAccessList using only the SID.	

](RS\_Diag\_04251)

### 8.3.1.29.9 diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function

[SWS\_DM\_01165]{DRAFT} Definition of API function ara::diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
<b>Scope:</b>	class ara::diag::DiagnosticServiceDynamicAccessList	
<b>Symbol:</b>	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::ByteString serviceHead)	
<b>Syntax:</b>	auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::ByteString serviceHead) noexcept -> DynamicAccessListDiagServiceBuilder;	
<b>Parameters (in):</b>	serviceHead	A string of bytes to start the DynamicAccess pattern-match
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder	An instance of a diagnostic service pattern builder
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the Application to construct a pattern for the DynamicAccessList using a string of bytes.	

](RS\_Diag\_04251)

### 8.3.1.30 ExternalAuthentication class

[SWS\_DM\_01191]{DRAFT} Definition of API class `ara::diag::ExternalAuthentication` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/external_authentication.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>ExternalAuthentication</code>
<b>Syntax:</b>	<code>class ExternalAuthentication final {...};</code>
<b>Description:</b>	Definition of the <code>ExternalAuthentication</code> class, which is used by the application to receive an instance of the <code>ClientAuthentication</code> Class relevant to the specific client.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.30.1 `diag::ExternalAuthentication::Address` type

[SWS\_DM\_01192]{DRAFT} Definition of API type `ara::diag::ExternalAuthentication::Address` [

<b>Kind:</b>	type alias
<b>Header file:</b>	<code>#include "ara/diag/external_authentication.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::ExternalAuthentication</code></a>
<b>Symbol:</b>	<code>Address</code>
<b>Syntax:</b>	<code>using Address = std::uint16_t;</code>
<b>Description:</b>	Alias for tester address.

]([RS\\_Diag\\_04251](#))

#### 8.3.1.30.2 `diag::ExternalAuthentication::ExternalAuthentication` constructor

[SWS\_DM\_01193]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::ExternalAuthentication` [

<b>Kind:</b>	function	
<b>Header file:</b>	<code>#include "ara/diag/external_authentication.h"</code>	
<b>Scope:</b>	<a href="#">class <code>ara::diag::ExternalAuthentication</code></a>	
<b>Symbol:</b>	<code>ExternalAuthentication(ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType)</code>	
<b>Syntax:</b>	<code>explicit ExternalAuthentication (ara::core::InstanceSpecifier instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;</code>	
<b>Parameters (in):</b>	<code>instanceSpecifier</code>	InstanceSpecifier to a PortPrototype of a DiagnosticAuthentication service instance in the manifest
	<code>reentrancyType</code>	specifies if interface is callable fully- or non-reentrant





<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Constructs the port for authentication of diagnostic clients.

]([RS\\_Diag\\_04251](#))

### 8.3.1.30.3 diag::ExternalAuthentication::ExternalAuthentication copy constructor

[SWS\_DM\_01196]{DRAFT} Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/external_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ExternalAuthentication</a>
<b>Symbol:</b>	ExternalAuthentication(ExternalAuthentication const &other)
<b>Syntax:</b>	ExternalAuthentication (ExternalAuthentication const &other)=delete;
<b>Description:</b>	Copy constructor of ExternalAuthentication.

]([RS\\_Diag\\_04251](#))

### 8.3.1.30.4 diag::ExternalAuthentication::ExternalAuthentication move constructor

[SWS\_DM\_01194]{DRAFT} Definition of API function ara::diag::ExternalAuthentication::ExternalAuthentication [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/external_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ExternalAuthentication</a>
<b>Symbol:</b>	ExternalAuthentication(ExternalAuthentication &&other)
<b>Syntax:</b>	ExternalAuthentication (ExternalAuthentication &&other) noexcept=default;
<b>Parameters (in):</b>	other   Object to move-construct from
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of ExternalAuthentication.

]([RS\\_Diag\\_04251](#))

### 8.3.1.30.5 diag::ExternalAuthentication::ExternalAuthentication copy assignment operator

[SWS\_DM\_01197]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/external_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ExternalAuthentication</a>
<b>Symbol:</b>	<code>operator=(ExternalAuthentication const &amp;other)</code>
<b>Syntax:</b>	<code>auto operator= (ExternalAuthentication const &amp;other) -&gt; ExternalAuthentication &amp;=delete;</code>
<b>Description:</b>	Copy assignment operator of ExternalAuthentication.

]([RS\\_Diag\\_04251](#))

### 8.3.1.30.6 diag::ExternalAuthentication::ExternalAuthentication move assignment operator

[SWS\_DM\_01195]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/external_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ExternalAuthentication</a>
<b>Symbol:</b>	<code>operator=(ExternalAuthentication &amp;&amp;other)</code>
<b>Syntax:</b>	<code>auto operator= (ExternalAuthentication &amp;&amp;other) &amp; noexcept -&gt; ExternalAuthentication &amp; =default;</code>
<b>Parameters (in):</b>	other   Object to move-assign from.
<b>Return value:</b>	ExternalAuthentication & =default   -
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move assignment operator of ExternalAuthentication.

]([RS\\_Diag\\_04251](#))

### 8.3.1.30.7 diag::ExternalAuthentication::~~ExternalAuthentication destructor

[SWS\_DM\_01198]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::~~ExternalAuthentication` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/external_authentication.h"
<b>Scope:</b>	<a href="#">class ara::diag::ExternalAuthentication</a>
<b>Symbol:</b>	<code>~ExternalAuthentication()</code>



△

<b>Syntax:</b>	<code>~ExternalAuthentication () noexcept;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DiagnosticServiceDynamicAccessList.

|(RS\_Diag\_04251)

### 8.3.1.30.8 diag::ExternalAuthentication::Get function

[SWS\_DM\_01199]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::Get` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/external_authentication.h"	
<b>Scope:</b>	class <a href="#">ara::diag::ExternalAuthentication</a>	
<b>Symbol:</b>	Get(const MetaInfo &metaInfo)	
<b>Syntax:</b>	<code>ara::core::Result&lt; ClientAuthentication &gt; Get (const MetaInfo &amp;metaInfo) noexcept;</code>	
<b>Parameters (in):</b>	metaInfo	The meta information of a diagnostic service port
<b>Return value:</b>	<code>ara::core::Result&lt; ClientAuthentication &gt;</code>	The diagnostic client associated authentication object or error
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	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.	

|(RS\_Diag\_04251)

### 8.3.1.30.9 diag::ExternalAuthentication::Get function

[SWS\_DM\_01200]{DRAFT} Definition of API function `ara::diag::ExternalAuthentication::Get` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/external_authentication.h"	
<b>Scope:</b>	class <a href="#">ara::diag::ExternalAuthentication</a>	
<b>Symbol:</b>	Get(Address sourceAddress)	
<b>Syntax:</b>	<code>ara::core::Result&lt; ClientAuthentication &gt; Get (Address sourceAddress) noexcept;</code>	
<b>Parameters (in):</b>	sourceAddress	The source address of the client
<b>Return value:</b>	<code>ara::core::Result&lt; ClientAuthentication &gt;</code>	The associated diagnostic client authentication object or error
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	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.	

|(RS\_Diag\_04251)

### 8.3.1.30.10 diag::ExternalAuthentication::GetAll function

[SWS\_DM\_01201]{DRAFT} Definition of API function ara::diag::ExternalAuthentication::GetAll [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/external_authentication.h"	
<b>Scope:</b>	class ara::diag::ExternalAuthentication	
<b>Symbol:</b>	GetAll()	
<b>Syntax:</b>	ara::core::Vector< ClientAuthentication > GetAll () noexcept;	
<b>Return value:</b>	ara::core::Vector< Client Authentication >	The list of all diagnostic client associated authentication objects or empty list if none available.
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to get all the ClientAuthentication Instances that are currently handled by the DM.	

](RS\_Diag\_04251)

### 8.3.1.31 DynamicAccessListDiagServiceBuilder class

[SWS\_DM\_01166]{DRAFT} Definition of API class ara::diag::DynamicAccessListDiagServiceBuilder [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	DynamicAccessListDiagServiceBuilder	
<b>Syntax:</b>	class DynamicAccessListDiagServiceBuilder final {...};	
<b>Description:</b>	Definition of the DynamicAccessListDiagServiceBuilder class, which is used by the application to build a DynamicAccessList.	

](RS\_Diag\_04251)

#### 8.3.1.31.1 ByteRange class

[SWS\_DM\_01182]{DRAFT} Definition of API class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder	
<b>Symbol:</b>	ByteRange	
<b>Syntax:</b>	class ByteRange final {...};	
<b>Description:</b>	Represents a single byte value closed range. Type alias of a single diagnostic service pattern element in the DynamicAccessList	

|(RS\_Diag\_04251)

### 8.3.1.31.1.1 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange constructor

[SWS\_DM\_01183]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	ByteRange(Byte min, Byte max)	
<b>Syntax:</b>	ByteRange (Byte min, Byte max) noexcept;	
<b>Parameters (in):</b>	min	The minimum value to match
	max	The maximum value to match
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructs a value range.	

|(RS\_Diag\_04251)

### 8.3.1.31.1.2 diag::DynamicAccessListDiagServiceBuilder::ByteRange::~~ByteRange destructor

[SWS\_DM\_01188]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::~~ByteRange [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	~ByteRange()	
<b>Syntax:</b>	~ByteRange () noexcept;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructor of ByteRange.	

|(RS\_Diag\_04251)



### 8.3.1.31.1.3 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy constructor

[SWS\_DM\_01184]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	ByteRange(ByteRange const &other)	
<b>Syntax:</b>	ByteRange (ByteRange const &other) noexcept;	
<b>DIRECTION NOT DEFINED</b>	other	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Copy constructor of ByteRange.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.1.4 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move constructor

[SWS\_DM\_01185]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	ByteRange(ByteRange &&other)	
<b>Syntax:</b>	ByteRange (ByteRange &&other) noexcept;	
<b>DIRECTION NOT DEFINED</b>	other	–
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of ByteRange.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.1.5 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy assignment operator

[SWS\_DM\_01186]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	operator=(ByteRange const &other)	
<b>Syntax:</b>	auto operator= (ByteRange const &other) & -> ByteRange &;	
<b>DIRECTION NOT DEFINED</b>	other	-
<b>Return value:</b>	ByteRange &	-
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Copy assignment operator of ByteRange.	

](RS\_Diag\_04251)

### 8.3.1.31.1.6 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move assignment operator

[SWS\_DM\_01187]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	operator=(ByteRange &&other)	
<b>Syntax:</b>	auto operator= (ByteRange &&other) & -> ByteRange &;	
<b>DIRECTION NOT DEFINED</b>	other	-
<b>Return value:</b>	ByteRange &	-
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Move assignment operator of ByteRange.	

](RS\_Diag\_04251)

### 8.3.1.31.1.7 diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin function

[SWS\_DM\_01189]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	GetMin()	
<b>Syntax:</b>	auto GetMin () const noexcept -> Byte;	
<b>Return value:</b>	Byte	The the lowest value
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Reports the lowest value to match.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.1.8 diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax function

[SWS\_DM\_01190]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
<b>Symbol:</b>	GetMax()	
<b>Syntax:</b>	auto GetMax () const noexcept -> Byte;	
<b>Return value:</b>	Byte	The the highest value
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Reports the highest value to match.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.2 diag::DynamicAccessListDiagServiceBuilder::Byte type

[SWS\_DM\_01167]{DRAFT} Definition of API type ara::diag::DynamicAccessListDiagServiceBuilder::Byte [

<b>Kind:</b>	type alias	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder	
<b>Symbol:</b>	Byte	



△

<b>Syntax:</b>	<code>using Byte = std::uint8_t;</code>
<b>Description:</b>	Type alias of a single diagnostic service pattern element in the DynamicAccessList.

 ] ([RS\\_Diag\\_04251](#))

### 8.3.1.31.3 diag::DynamicAccessListDiagServiceBuilder::ByteString type

[SWS\_DM\_01168]{DRAFT} Definition of API type `ara::diag::DynamicAccessListDiagServiceBuilder::ByteString` [

<b>Kind:</b>	type alias
<b>Header file:</b>	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
<b>Scope:</b>	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
<b>Symbol:</b>	ByteString
<b>Syntax:</b>	<code>using ByteString = ara::core::Span&lt;Byte&gt;;</code>
<b>Description:</b>	Type alias of a sequence of diagnostic service pattern elements in the DynamicAccessList.

 ] ([RS\\_Diag\\_04251](#))

### 8.3.1.31.4 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder constructor

[SWS\_DM\_01169]{DRAFT} Definition of API function `ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder` [

<b>Kind:</b>	function	
<b>Header file:</b>	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>	
<b>Scope:</b>	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>	
<b>Symbol:</b>	<code>DynamicAccessListDiagServiceBuilder(V value, ara::core::Vector&lt; std::uint8_t &gt; &amp;content)</code>	
<b>Syntax:</b>	<pre>template &lt;typename V&gt; DynamicAccessListDiagServiceBuilder (V value, ara::core::Vector&lt; std::uint8_t &gt; &amp;content) noexcept;</pre>	
<b>Parameters (in):</b>	value	value(s) to be added to the DynamicAccessList
	content	Stack holder for serialized DynamicAccessList
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructor for DynamicAccessListDiagServiceBuilder.	

 ] ([RS\\_Diag\\_04251](#))

### 8.3.1.31.5 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy constructor

[SWS\_DM\_01171]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>
<b>Symbol:</b>	DynamicAccessListDiagServiceBuilder(DynamicAccessListDiagServiceBuilder const &other)
<b>Syntax:</b>	DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagServiceBuilder const &other)=delete;
<b>Description:</b>	Copy constructor of DynamicAccessListDiagServiceBuilder cannot be used.

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.6 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move constructor

[SWS\_DM\_01170]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>
<b>Symbol:</b>	DynamicAccessListDiagServiceBuilder(DynamicAccessListDiagServiceBuilder &&other)
<b>Syntax:</b>	DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagServiceBuilder &&other) noexcept;
<b>Parameters (in):</b>	other      Object to move-construct from
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of DynamicAccessListDiagServiceBuilder.

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.7 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy assignment operator

[SWS\_DM\_01172]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>
<b>Symbol:</b>	operator=(DynamicAccessListDiagServiceBuilder const &other)





<b>Syntax:</b>	<code>auto operator= (DynamicAccessListDiagServiceBuilder const &amp;other) -&gt; DynamicAccessListDiagServiceBuilder &amp;=delete;</code>
<b>Description:</b>	Copy assignment operator of DynamicAccessListDiagServiceBuilder cannot be used.

|(RS\_Diag\_04251)

### 8.3.1.31.8 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move assignment operator

[SWS\_DM\_01173]{DRAFT} Definition of API function `ara::diag::DynamicAccessListDiagServiceBuilder::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
<b>Scope:</b>	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
<b>Symbol:</b>	<code>operator=(DynamicAccessListDiagServiceBuilder &amp;&amp;other)</code>
<b>Syntax:</b>	<code>auto operator= (DynamicAccessListDiagServiceBuilder &amp;&amp;other) -&gt; DynamicAccessListDiagServiceBuilder &amp;=delete;</code>
<b>Description:</b>	Move assignment operator of DynamicAccessListDiagServiceBuilder.

|(RS\_Diag\_04251)

### 8.3.1.31.9 diag::DynamicAccessListDiagServiceBuilder::~DynamicAccessListDiagServiceBuilder destructor

[SWS\_DM\_01174]{DRAFT} Definition of API function `ara::diag::DynamicAccessListDiagServiceBuilder::~DynamicAccessListDiagServiceBuilder` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
<b>Scope:</b>	<code>class ara::diag::DynamicAccessListDiagServiceBuilder</code>
<b>Symbol:</b>	<code>~DynamicAccessListDiagServiceBuilder()</code>
<b>Syntax:</b>	<code>~DynamicAccessListDiagServiceBuilder () noexcept;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DynamicAccessListDiagServiceBuilder.

|(RS\_Diag\_04251)

### 8.3.1.31.10 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS\_DM\_01175]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>	
<b>Symbol:</b>	Add(Byte value)	
<b>Syntax:</b>	auto Add (Byte value) noexcept -> DynamicAccessListDiagServiceBuilder &;	
<b>Parameters (in):</b>	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
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to add a single byte to a DynamicAccessListPattern.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.11 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS\_DM\_01176]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>	
<b>Symbol:</b>	Add(ByteString values)	
<b>Syntax:</b>	auto Add (ByteString values) noexcept -> DynamicAccessListDiagServiceBuilder &;	
<b>Parameters (in):</b>	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
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to add a string of bytes to a DynamicAccessListPattern.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.12 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS\_DM\_01177]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Add [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>	
<b>Symbol:</b>	Add(ByteRange range)	
<b>Syntax:</b>	auto Add (ByteRange range) noexcept -> DynamicAccessListDiagServiceBuilder &;	
<b>Parameters (in):</b>	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
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to add a range of bytes to a DynamicAccessListPattern.	

]([RS\\_Diag\\_04251](#))

### 8.3.1.31.13 diag::DynamicAccessListDiagServiceBuilder::Any function

[SWS\_DM\_01178]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Any [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>	
<b>Symbol:</b>	Any(std::size_t numberOfBytesToIgnore)	
<b>Syntax:</b>	auto Any (std::size_t numberOfBytesToIgnore) noexcept -> DynamicAccessListDiagServiceBuilder &;	
<b>Parameters (in):</b>	numberOfBytesToIgnore	The number of bytes to ignore
<b>Return value:</b>	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04251](#))



### 8.3.1.31.14 diag::DynamicAccessListDiagServiceBuilder::EndsWith function

[SWS\_DM\_01179]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::EndsWith [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder	
<b>Symbol:</b>	EndsWith(Byte value)	
<b>Syntax:</b>	void EndsWith (Byte value) noexcept;	
<b>Parameters (in):</b>	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
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to specify a single byte value at the end of the DynamicAccessList pattern.	

](RS\_Diag\_04251)

### 8.3.1.31.15 diag::DynamicAccessListDiagServiceBuilder::EndsWith function

[SWS\_DM\_01180]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::EndsWith [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
<b>Scope:</b>	class ara::diag::DynamicAccessListDiagServiceBuilder	
<b>Symbol:</b>	EndsWith(ByteRange range)	
<b>Syntax:</b>	void EndsWith (ByteRange range) noexcept;	
<b>Parameters (in):</b>	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
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	This function is used by the application to specify a closed range of bytes at the end of the DynamicAccessList pattern.	

](RS\_Diag\_04251)

### 8.3.1.31.16 diag::DynamicAccessListDiagServiceBuilder::Build function

[SWS\_DM\_01181]{DRAFT} Definition of API function ara::diag::DynamicAccessListDiagServiceBuilder::Build [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
<b>Scope:</b>	<a href="#">class ara::diag::DynamicAccessListDiagServiceBuilder</a>
<b>Symbol:</b>	Build()
<b>Syntax:</b>	void Build () noexcept;
<b>Return value:</b>	None
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Finalizes building the DynamicAccessList. Must be called before destroying the ServiceBuilder object.

]([RS\\_Diag\\_04251](#))

### 8.3.1.32 FileTransferService class

[SWS\_DM\_01320]{DRAFT} Definition of API class ara::diag::FileTransferService [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	FileTransferService
<b>Syntax:</b>	class FileTransferService {...};
<b>Description:</b>	File Transfer service interface.

]([RS\\_Diag\\_04135](#))

#### 8.3.1.32.1 diag::FileTransferService::FileSizes type

[SWS\_DM\_01321]{DRAFT} Definition of API class ara::diag::FileTransferService::FileSizes [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>
<b>Symbol:</b>	FileSizes
<b>Syntax:</b>	struct FileSizes {...};
<b>Description:</b>	Definition of total file sizes .

]([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01322]{DRAFT} Definition of API variable ara::diag::FileTransferService::FileSizes::uncompressed\_size** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Scope:</b>	<a href="#">struct ara::diag::FileTransferService::FileSizes</a>
<b>Symbol:</b>	uncompressed_size
<b>Type:</b>	std::uint64_t
<b>Syntax:</b>	std::uint64_t uncompressed_size;
<b>Description:</b>	Specifies the uncompressed file size in bytes .

]([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01323]{DRAFT} Definition of API variable ara::diag::FileTransferService::FileSizes::compressed\_size** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Scope:</b>	<a href="#">struct ara::diag::FileTransferService::FileSizes</a>
<b>Symbol:</b>	compressed_size
<b>Type:</b>	std::uint64_t
<b>Syntax:</b>	std::uint64_t compressed_size;
<b>Description:</b>	Specifies the compressed file size in bytes .

]([RS\\_Diag\\_04135](#))

### 8.3.1.32.2 diag::FileTransferService::WriteFileMode enum

**[SWS\_DM\_01324]{DRAFT} Definition of API enum ara::diag::FileTransferService::WriteFileMode** [

<b>Kind:</b>	enumeration				
<b>Header file:</b>	#include "ara/diag/file_transfer.h"				
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"				
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>				
<b>Symbol:</b>	WriteFileMode				
<b>Underlying type:</b>	ara::core::Byte				
<b>Syntax:</b>	enum class WriteFileMode : ara::core::Byte {...};				
<b>Values:</b>	<table border="1"> <tr> <td>kAdd</td> <td>The file shall be added only if not existing.</td> </tr> <tr> <td>kReplace</td> <td>The file shall be added in any case, if already exists, will be replaced</td> </tr> </table>	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
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				
<b>Description:</b>	Determines the write file operation mode .				

]([RS\\_Diag\\_04135](#))

### 8.3.1.32.3 diag::FileTransferService::FileTransferService constructors

#### [SWS\_DM\_01325]{DRAFT} Definition of API function ara::diag::FileTransferService::FileTransferService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>	
<b>Symbol:</b>	FileTransferService(ara::core::InstanceSpecifier const &instanceSpecifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	FileTransferService (ara::core::InstanceSpecifier const &instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to an PortPrototype of an DownloadService Interface
	reentrancyType	Specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructor for FileTransferService (inherited)	

]([RS\\_Diag\\_04135](#))

#### [SWS\_DM\_01327]{DRAFT} Definition of API function ara::diag::FileTransferService::FileTransferService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>	
<b>Symbol:</b>	FileTransferService(FileTransferService const &)	
<b>Syntax:</b>	FileTransferService (FileTransferService const &)=delete;	
<b>Description:</b>	FileTransferService shall be a single not copy-able instance. .	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04135](#))

#### [SWS\_DM\_01328]{DRAFT} Definition of API function ara::diag::FileTransferService::FileTransferService [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>	
<b>Symbol:</b>	FileTransferService(FileTransferService &&other)	
<b>Syntax:</b>	FileTransferService (FileTransferService &&other) noexcept;	
<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructs an instance of FileTransferService.	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04135](#))

### 8.3.1.32.4 diag::FileTransferService::~~FileTransferService destructor

[SWS\_DM\_01326]{DRAFT} Definition of API function ara::diag::FileTransferService::~~FileTransferService [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Scope:</b>	class ara::diag::FileTransferService
<b>Symbol:</b>	~FileTransferService()
<b>Syntax:</b>	virtual ~FileTransferService () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class FileTransferService.

](RS\_Diag\_04135)

### 8.3.1.32.5 diag::FileTransferService::FileTransferService copy assignment operator

[SWS\_DM\_01329]{DRAFT} Definition of API function ara::diag::FileTransferService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Scope:</b>	class ara::diag::FileTransferService
<b>Symbol:</b>	operator=(FileTransferService const &)
<b>Syntax:</b>	auto operator= (FileTransferService const &) -> FileTransferService &=delete;
<b>Description:</b>	FileTransferService shall be a single not assignable instance.
<b>Visibility:</b>	protected

](RS\_Diag\_04135)

### 8.3.1.32.6 diag::FileTransferService::FileTransferService move assignment operator

[SWS\_DM\_01330]{DRAFT} Definition of API function ara::diag::FileTransferService::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/file_transfer.h"
<b>Scope:</b>	class ara::diag::FileTransferService
<b>Symbol:</b>	operator=(FileTransferService &&other)
<b>Syntax:</b>	auto operator= (FileTransferService &&other) & noexcept -> FileTransferService &;





<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	FileTransferService &	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of FileTransferService.	
<b>Visibility:</b>	protected	

|(RS\_Diag\_04135)

### 8.3.1.32.7 diag::FileTransferService::RequestReadFile function

[SWS\_DM\_01331]{DRAFT} Definition of API function ara::diag::FileTransferService::RequestReadFile [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	class ara::diag::FileTransferService	
<b>Symbol:</b>	RequestReadFile(ara::core::String fileName, ara::core::Byte dataFormatIdentifier, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual auto RequestReadFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, MetaInfo const &amp;metaInfo, CancellationHandler cancellationHandler) noexcept -&gt; ara::core::Future&lt; std::tuple&lt; DataTransferReadSession, FileSizes &gt; &gt;=0;</pre>	
<b>Parameters (in):</b>	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< std::tuple< DataTransferReadSession, FileSizes > >=0	A future with either the access strategy to be used during reading and the file sizes to be reported to the UDS client for a positive response message) or an UDS NRC value (for a negative response message) uncompressed size: If set to 0 - the parameter will not be reported to the diagnostic client compressed size: If set to 0 - the parameter will not be reported to the diagnostic client. If no compression is used, the value shall be zero.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Called for RequestFileTransfer with ModeOfOperation ReadFile.	

|(RS\_Diag\_04135)

### 8.3.1.32.8 diag::FileTransferService::RequestReadDirectory function

[SWS\_DM\_01332]{DRAFT} Definition of API function ara::diag::FileTransferService::RequestReadDirectory [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	class ara::diag::FileTransferService	
<b>Symbol:</b>	RequestReadDirectory(ara::core::String directoryName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual auto RequestReadDirectory (ara::core::String directoryName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< std::tuple< DataTransferReadSession, std::uint64_t > >=0;	
<b>Parameters (in):</b>	directoryName	Path including name of the directory to read
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< std::tuple< DataTransferReadSession, std::uint64_t > >=0	A future with either the access strategy to be used during reading and the directory information size to be reported to the UDS client for a positive response message) or an UDS NRC value (for a negative response message)
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Called for RequestFileTransfer with ModeOfOperation ReadDir.	

] ([RS\\_Diag\\_04135](#))

### 8.3.1.32.9 diag::FileTransferService::RequestWriteFile function

[SWS\_DM\_01333]{DRAFT} Definition of API function ara::diag::FileTransferService::RequestWriteFile [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	class ara::diag::FileTransferService	
<b>Symbol:</b>	RequestWriteFile(ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes file Sizes, WriteFileMode mode, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual auto RequestWriteFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes fileSizes, WriteFileMode mode, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< DataTransferWriteSession >=0;	
<b>Parameters (in):</b>	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier





	fileSizes	The compressed/uncompressed files sizes
	mode	The file replacement mode
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< Data TransferWriteSession > >=0	A future with either an instance of the file writing session or an UDS NRC value (for an negative response message) A future with either an instance of the file writing session or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Called for RequestFileTransfer with ModeOfOperation WriteFile.	

] ([RS\\_Diag\\_04135](#))

### 8.3.1.32.10 diag::FileTransferService::RequestResumeWriteFile function

[SWS\_DM\_01334]{DRAFT} Definition of API function ara::diag::FileTransferService::RequestResumeWriteFile [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>	
<b>Symbol:</b>	RequestResumeWriteFile(ara::core::String fileName, ara::core::Byte dataFormatIdentifier, File Sizes fileSizes, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual auto RequestResumeWriteFile (ara::core::String fileName, ara::core::Byte dataFormatIdentifier, FileSizes fileSizes, MetaInfo const &amp;metaInfo, CancellationHandler cancellationHandler) noexcept -&gt; ara::core::Future&lt; std::tuple&lt; DataTransferWriteSession, std::uint64_t &gt; &gt;=0;</pre>	
<b>Parameters (in):</b>	fileName	Path including name of the file to read
	dataFormatIdentifier	UDS dataFormat Identifier
	fileSizes	The compressed/uncompressed files sizes
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< std::tuple< DataTransfer WriteSession, std::uint64_t > >=0	A future with either an instance of the file writing session and the byte position the client shall start resuming from or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	noexcept	







<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Called for RequestFileTransfer with ModeOfOperation ResumeFile.	

](RS\_Diag\_04135)

### 8.3.1.32.11 diag::FileTransferService::DeleteFile function

[SWS\_DM\_01335][DRAFT] Definition of API function ara::diag::FileTransferService::DeleteFile [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	class ara::diag::FileTransferService	
<b>Symbol:</b>	DeleteFile(ara::core::String fileName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual auto DeleteFile (ara::core::String fileName, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept -> ara::core::Future< void >=0;	
<b>Parameters (in):</b>	fileName	Path including name of the file to read
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >=0	A future with either a void data result or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Called for RequestFileTransfer with ModeOfOperation DeleteFile. This method is the complete operation of deleting a file	

](RS\_Diag\_04135)

### 8.3.1.32.12 diag::FileTransferService::Offer function

[SWS\_DM\_01336][DRAFT] Definition of API function ara::diag::FileTransferService::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/file_transfer.h"	
<b>Scope:</b>	class ara::diag::FileTransferService	
<b>Symbol:</b>	Offer()	





<b>Syntax:</b>	<code>auto Offer () noexcept -&gt; ara::core::Result&lt; void &gt;;</code>	
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	Positive result if service was offered, error if offer failed
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Errors:</b>	<code>DiagLibErrc::kConfigurationMismatch</code>	if handle is an invalid handle
	<code>DiagLibErrc::kAlreadyOffered</code>	if service has been offered already
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.32.13 diag::FileTransferService::StopOffer function

[SWS\_DM\_01337]{DRAFT} Definition of API function `ara::diag::FileTransferService::StopOffer` [

<b>Kind:</b>	function	
<b>Header file:</b>	<code>#include "ara/diag/file_transfer.h"</code>	
<b>Scope:</b>	<a href="#">class ara::diag::FileTransferService</a>	
<b>Symbol:</b>	<code>StopOffer()</code>	
<b>Syntax:</b>	<code>void StopOffer () const noexcept;</code>	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.32.14 diag::DataTransferExitType enum

[SWS\_DM\_01538]{DRAFT} Definition of API enum `ara::diag::DataTransferExitType` [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	<code>#include "ara/diag/data_transfer.h"</code>	
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>DataTransferExitType</code>	
<b>Underlying type:</b>	<code>ara::core::Byte</code>	
<b>Syntax:</b>	<code>enum class DataTransferExitType : ara::core::Byte {...};</code>	
<b>Values:</b>	<code>kkAcknowledge</code>	The file transfer finished.
	<code>kAbort</code>	The file transfer has been aborted.





<b>Description:</b>	Determines transfer exit signal type.
---------------------	---------------------------------------

]([RS\\_Diag\\_04135](#))

### 8.3.1.33 DataTransferWriteHandler class

[SWS\_DM\_01539]{DRAFT} Definition of API class ara::diag::DataTransferWriteHandler [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DataTransferWriteHandler
<b>Syntax:</b>	class DataTransferWriteHandler {...};
<b>Description:</b>	Handles data transfers initiated by AddFile, ResumeFile or ReplaceFile ModeOfOperation handling per requested RequestWriteFile.

]([RS\\_Diag\\_04135](#))

#### 8.3.1.33.1 diag::DataTransferWriteHandler::DataTransferWriteHandler constructors

[SWS\_DM\_01540]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferWriteHandler</a>
<b>Symbol:</b>	DataTransferWriteHandler()
<b>Syntax:</b>	DataTransferWriteHandler ()=default;
<b>Description:</b>	Constructs an instance of this class.
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04135](#))

[SWS\_DM\_01542]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferWriteHandler</a>
<b>Symbol:</b>	DataTransferWriteHandler(DataTransferWriteHandler &&other)





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

](RS\_Diag\_04135)

### [SWS\_DM\_01544]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::DataTransferWriteHandler [

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

](RS\_Diag\_04135)

### 8.3.1.33.2 diag::DataTransferWriteHandler::~~DataTransferWriteHandler destructor

### [SWS\_DM\_01541]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::~~DataTransferWriteHandler [

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

](RS\_Diag\_04135)

### 8.3.1.33.3 diag::DataTransferWriteHandler::DataTransferWriteHandler copy assignment operator

[SWS\_DM\_01545]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::operator= [

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

](RS\_Diag\_04135)

### 8.3.1.33.4 diag::DataTransferWriteHandler::DataTransferWriteHandler move assignment operator

[SWS\_DM\_01543]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferWriteHandler	
<b>Symbol:</b>	operator=(DataTransferWriteHandler &&other)	
<b>Syntax:</b>	auto operator= (DataTransferWriteHandler &&other) & noexcept -> DataTransferWriteHandler & =default;	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferWriteHandler & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

](RS\_Diag\_04135)

### 8.3.1.33.5 diag::DataTransferWriteHandler::Write function

[SWS\_DM\_01546]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::Write [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	class ara::diag::DataTransferWriteHandler





<b>Symbol:</b>	Write(ara::core::Span< ara::core::Byte > requestData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual auto Write (ara::core::Span&lt; ara::core::Byte &gt; requestData, MetaInfo const &amp;metaInfo, CancellationHandler cancellationHandler) noexcept -&gt; ara::core::Future&lt; void &gt;=0;</pre>	
<b>Parameters (in):</b>	requestData	The first or subsequent file content chunk received from the UDS client to be written into the file.
	metaInfo	Contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< void >=0	A Future, which either is of type void or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message).
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrcErrc
<b>Description:</b>	Appends a data chunk into application memory.	

](RS\_Diag\_04135)

### 8.3.1.33.6 diag::DataTransferWriteHandler::ExitWrite function

[SWS\_DM\_01547]{DRAFT} Definition of API function ara::diag::DataTransferWriteHandler::ExitWrite [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferWriteHandler	
<b>Symbol:</b>	ExitWrite(DataTransferExitType exitType, ara::core::Span< ara::core::Byte > transferRequestParameterRecord, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual auto ExitWrite (DataTransferExitType exitType, ara::core::Span&lt; ara::core::Byte &gt; transferRequestParameterRecord, MetaInfo const &amp;metaInfo, CancellationHandler cancellationHandler) noexcept -&gt; ara::core::Future&lt; ara::core::Vector&lt; ara::core::Byte &gt; &gt;=0;</pre>	
<b>Parameters (in):</b>	exitType	Specifies the exit reason
	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.
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).





<b>Return value:</b>	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_data will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Exits an ongoing data transfer session.	

|(RS\_Diag\_04135)

### 8.3.1.34 DataTransferReadSharedDataHandler class

[SWS\_DM\_01497]{DRAFT} **Definition of API class ara::diag::DataTransferReadSharedDataHandler** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DataTransferReadSharedDataHandler
<b>Syntax:</b>	class DataTransferReadSharedDataHandler {...};
<b>Description:</b>	Handles data transfers initiated by RequestReadFile/-Directory with shared data from the implementation.

|(RS\_Diag\_04135)

#### 8.3.1.34.1 diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler constructor

[SWS\_DM\_01498]{DRAFT} **Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSharedDataHandler</a>
<b>Symbol:</b>	DataTransferReadSharedDataHandler()
<b>Syntax:</b>	DataTransferReadSharedDataHandler () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Constructs an instance of this class.

|(RS\_Diag\_04135)

### 8.3.1.34.2 diag::DataTransferReadSharedDataHandler::DataTransferRead-SharedDataHandler move constructor

[SWS\_DM\_01500]{DRAFT} Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadSharedDataHandler	
<b>Symbol:</b>	DataTransferReadSharedDataHandler(DataTransferReadSharedDataHandler &&other)	
<b>Syntax:</b>	DataTransferReadSharedDataHandler (DataTransferReadSharedDataHandler &&other) noexcept=default;	
<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructs an instance of this class.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.34.3 diag::DataTransferReadSharedDataHandler::DataTransferRead-SharedDataHandler copy constructor

[SWS\_DM\_01501]{DRAFT} Definition of API function ara::diag::DataTransferReadSharedDataHandler::DataTransferReadSharedDataHandler [

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

]([RS\\_Diag\\_04135](#))

### 8.3.1.34.4 diag::DataTransferReadSharedDataHandler::~~DataTransferRead-SharedDataHandler destructor

[SWS\_DM\_01499]{DRAFT} Definition of API function ara::diag::DataTransferReadSharedDataHandler::~~DataTransferReadSharedDataHandler [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadSharedDataHandler	
<b>Symbol:</b>	~DataTransferReadSharedDataHandler()	







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

](RS\_Diag\_04135)

### 8.3.1.34.5 diag::DataTransferReadSharedDataHandler::DataTransferRead-SharedDataHandler move assignment operator

[SWS\_DM\_01502]{DRAFT} Definition of API function `ara::diag::DataTransferReadSharedDataHandler::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSharedDataHandler</a>	
<b>Symbol:</b>	<code>operator=(DataTransferReadSharedDataHandler &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>auto operator= (DataTransferReadSharedDataHandler &amp;&amp;other) &amp; noexcept -&gt; DataTransferReadSharedDataHandler &amp; =default;</code>	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferRead SharedDataHandler & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

](RS\_Diag\_04135)

### 8.3.1.34.6 diag::DataTransferReadSharedDataHandler::DataTransferRead-SharedDataHandler copy assignment operator

[SWS\_DM\_01503]{DRAFT} Definition of API function `ara::diag::DataTransferReadSharedDataHandler::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSharedDataHandler</a>	
<b>Symbol:</b>	<code>operator=(DataTransferReadSharedDataHandler const &amp;)</code>	
<b>Syntax:</b>	<code>auto operator= (DataTransferReadSharedDataHandler const &amp;) -&gt; Data TransferReadSharedDataHandler &amp;=delete;</code>	
<b>Description:</b>	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .	

](RS\_Diag\_04135)

### 8.3.1.34.7 diag::DataTransferReadSharedDataHandler::Read function

#### [SWS\_DM\_01504]{DRAFT} Definition of API function ara::diag::DataTransferReadSharedDataHandler::Read [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadSharedDataHandler	
<b>Symbol:</b>	Read(ReleaseHandler release_handler, MetaInfo const &meta_info, CancellationHandler cancellation_handler)	
<b>Syntax:</b>	virtual auto Read (ReleaseHandler release_handler, MetaInfo const &meta_info, CancellationHandler cancellation_handler) noexcept -> ara::core::Future< ara::core::Span< ara::core::Byte > >=0;	
<b>Parameters (in):</b>	release_handler	The release shared resource handler to be stored and used by the application until a new one is passed for the same meta_info (UDS client)
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< ara::core::Span< ara::core::Byte > >=0	A Future, which either gets readied to ReadSharedOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for a negative response message). Data in ReleaseSharedHandler.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrcErrc
<b>Description:</b>	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.	

](RS\_Diag\_04135)

### 8.3.1.34.8 diag::DataTransferReadSharedDataHandler::ExitRead function

#### [SWS\_DM\_01505]{DRAFT} Definition of API function ara::diag::DataTransferReadSharedDataHandler::ExitRead [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadSharedDataHandler	
<b>Symbol:</b>	ExitRead(DataTransferExitType exit_type, ara::core::Span< ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler)	





<b>Syntax:</b>	<pre>virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span&lt; ara::core::Byte &gt; transfer_request_parameter_record, MetaInfo const &amp;meta_info, CancellationHandler cancellation_handler) noexcept -&gt; ara::core::Future&lt; ara::core::Vector&lt; ara::core::Byte &gt; &gt;=0;</pre>	
<b>Parameters (in):</b>	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.
	meta_info	contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	<pre>ara::core::Future&lt; ara::core::Vector&lt; ara::core::Byte &gt; &gt;=0</pre>	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_data will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrcErrc
<b>Description:</b>	Exits an ongoing data transfer session.	

|(RS\_Diag\_04135)

### 8.3.1.35 DataTransferReadByPullHandler class

[SWS\_DM\_01506][DRAFT] Definition of API class ara::diag::DataTransferReadByPullHandler [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DataTransferReadByPullHandler
<b>Syntax:</b>	<pre>class DataTransferReadByPullHandler {...};</pre>
<b>Description:</b>	Handles data transfers initiated by RequestReadFile/-Directory with data pulled by AraDiag from the implementation.

|(RS\_Diag\_04135)

### 8.3.1.35.1 diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler constructor

[SWS\_DM\_01507]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler` [

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

]([RS\\_Diag\\_04135](#))

### 8.3.1.35.2 diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler move constructor

[SWS\_DM\_01508]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadByPullHandler</a>
<b>Symbol:</b>	DataTransferReadByPullHandler(DataTransferReadByPullHandler &&other)
<b>Syntax:</b>	DataTransferReadByPullHandler (DataTransferReadByPullHandler &&other) noexcept=default;
<b>Parameters (out):</b>	other   The other object
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructs an instance of this class.

]([RS\\_Diag\\_04135](#))

### 8.3.1.35.3 diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler copy constructor

[SWS\_DM\_01509]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadByPullHandler</a>





<b>Symbol:</b>	DataTransferReadByPullHandler(DataTransferReadByPullHandler const &)
<b>Syntax:</b>	DataTransferReadByPullHandler (DataTransferReadByPullHandler const &)=delete;
<b>Description:</b>	Handlers shall not be copiable since only one way usage i.e. per single file transfer session .

|(RS\_Diag\_04135)

### 8.3.1.35.4 diag::DataTransferReadByPullHandler::~DataTransferReadByPullHandler destructor

[SWS\_DM\_01510]{DRAFT} Definition of API function ara::diag::DataTransferReadByPullHandler::~DataTransferReadByPullHandler [

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

|(RS\_Diag\_04135)

### 8.3.1.35.5 diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler move assignment operator

[SWS\_DM\_01511]{DRAFT} Definition of API function ara::diag::DataTransferReadByPullHandler::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadByPullHandler	
<b>Symbol:</b>	operator=(DataTransferReadByPullHandler &&other)	
<b>Syntax:</b>	auto operator= (DataTransferReadByPullHandler &&other) & noexcept -> DataTransferReadByPullHandler & =default;	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferReadByPullHandler & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

|(RS\_Diag\_04135)

### 8.3.1.35.6 diag::DataTransferReadByPullHandler::DataTransferReadByPullHandler copy assignment operator

[SWS\_DM\_01512]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPullHandler::operator=` [

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

]([RS\\_Diag\\_04135](#))

### 8.3.1.35.7 diag::DataTransferReadByPullHandler::Read function

[SWS\_DM\_01513]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPullHandler::Read` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class <code>ara::diag::DataTransferReadByPullHandler</code>	
<b>Symbol:</b>	<code>Read(ara::core::Span&lt; ara::core::Byte &gt; response_data, MetaInfo const &amp;meta_info, CancellationHandler cancellation_handler)</code>	
<b>Syntax:</b>	<code>virtual auto Read (ara::core::Span&lt; ara::core::Byte &gt; response_data, MetaInfo const &amp;meta_info, CancellationHandler cancellation_handler) noexcept -&gt; ara::core::Future&lt; std::uint32_t &gt;=0;</code>	
<b>Parameters (in):</b>	response_data	The view over a pre-allocated by AraDiag memory to write in this data chunk.
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling <code>cancellationHandler.IsCanceled()</code> (which 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).
<b>Return value:</b>	<code>ara::core::Future&lt; std::uint32_t &gt;=0</code>	A Future, which either gets readied to actual amount of data provided in the Span (for a positive response message) or readied with <code>ErrorCode</code> from <code>DiagUdsNrcErrc</code> (for an negative response message). Data in <code>response_data</code> will be placed after block <code>SequenceCounter</code> as <code>transferResponseParameterRecord</code> in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from <code>DiagUdsNrcErrc</code>





<b>Description:</b>	<p>Reads data chunk from the application.</p> <p>The first or subsequent file/directory info content chunk to be sent back to the UDS client will be composed by the application and copied into the provided by AraDiag Span with up to the requested Span size.</p>
---------------------	---

|(RS\_Diag\_04135)

### 8.3.1.35.8 diag::DataTransferReadByPullHandler::ExitRead function

#### [SWS\_DM\_01514]{DRAFT} Definition of API function ara::diag::DataTransferReadByPullHandler::ExitRead [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadByPullHandler	
<b>Symbol:</b>	ExitRead(DataTransferExitType exit_type, ara::core::Span< ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler)	
<b>Syntax:</b>	<pre>virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span&lt; ara::core::Byte &gt; transfer_request_parameter_record, MetaInfo const &amp;meta_info, CancellationHandler cancellation_handler) noexcept -&gt; ara::core::Future&lt; ara::core::Vector&lt; ara::core::Byte &gt; &gt;=0;</pre>	
<b>Parameters (in):</b>	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.
	meta_info	contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_data will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Exits an ongoing data transfer session.	

|(RS\_Diag\_04135)

### 8.3.1.36 DataTransferReadByPushHandler class

[SWS\_DM\_01515]{DRAFT} Definition of API class `ara::diag::DataTransferReadByPushHandler` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/data_transfer.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DataTransferReadByPushHandler</code>
<b>Syntax:</b>	<code>class DataTransferReadByPushHandler {...};</code>
<b>Description:</b>	Handles data transfers initiated by RequestReadFile/-Directory with data pushed by the implementation.

]([RS\\_Diag\\_04135](#))

#### 8.3.1.36.1 `diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler` constructor

[SWS\_DM\_01516]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler` [

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

]([RS\\_Diag\\_04135](#))

#### 8.3.1.36.2 `diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler` move constructor

[SWS\_DM\_01517]{DRAFT} Definition of API function `ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/data_transfer.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::DataTransferReadByPushHandler</code></a>
<b>Symbol:</b>	<code>DataTransferReadByPushHandler(DataTransferReadByPushHandler &amp;&amp;other)</code>
<b>Syntax:</b>	<code>DataTransferReadByPushHandler (DataTransferReadByPushHandler &amp;&amp;other) noexcept=default;</code>







<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructs an instance of this class.	

|(RS\_Diag\_04135)

### 8.3.1.36.3 diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler copy constructor

[SWS\_DM\_01518]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler [

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

|(RS\_Diag\_04135)

### 8.3.1.36.4 diag::DataTransferReadByPushHandler::~~DataTransferReadByPushHandler destructor

[SWS\_DM\_01519]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::~~DataTransferReadByPushHandler [

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

|(RS\_Diag\_04135)

### 8.3.1.36.5 diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler move assignment operator

[SWS\_DM\_01520]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadByPushHandler	
<b>Symbol:</b>	operator=(DataTransferReadByPushHandler &&other)	
<b>Syntax:</b>	auto operator= (DataTransferReadByPushHandler &&other) & noexcept -> DataTransferReadByPushHandler & =default;	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferReadByPushHandler & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

](RS\_Diag\_04135)

### 8.3.1.36.6 diag::DataTransferReadByPushHandler::DataTransferReadByPushHandler copy assignment operator

[SWS\_DM\_01521]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::operator= [

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

](RS\_Diag\_04135)

### 8.3.1.36.7 diag::DataTransferReadByPushHandler::Read function

[SWS\_DM\_01522]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::Read [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadByPushHandler	





<b>Symbol:</b>	Read(std::size_t recommended_number_bytes_to_return, MetaInfo const &meta_info, CancellationHandler cancellation_handler)	
<b>Syntax:</b>	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;	
<b>Parameters (in):</b>	recommended_number_bytes_to_return	The recommended number of bytes to send back to DM in order to get optimum data throughput
	meta_info	Contains additional meta information
	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< ara::core::Vector< ara::core::Byte > >=>0	A Future, which either gets readied to ReadPushOutput (transfer ResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message). Data in ReadPushOutput.responseData will be placed after blockSequenceCounter as transferResponse ParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrc Errc
<b>Description:</b>	Reads data chunk from the application.  The first or subsequent file/directory info content chunk to be sent back to the UDS client will be composed by the application and its ownership is passed as a result to AraDiag.	

](RS\_Diag\_04135)

### 8.3.1.36.8 diag::DataTransferReadByPushHandler::ExitRead function

[SWS\_DM\_01523]{DRAFT} Definition of API function ara::diag::DataTransferReadByPushHandler::ExitRead [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadByPushHandler	
<b>Symbol:</b>	ExitRead(DataTransferExitType exit_type, ara::core::Span< ara::core::Byte > transfer_request_parameter_record, MetaInfo const &meta_info, CancellationHandler cancellation_handler)	
<b>Syntax:</b>	virtual auto ExitRead (DataTransferExitType exit_type, ara::core::Span< 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;	
<b>Parameters (in):</b>	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.
	meta_info	contains additional meta information





	cancellation_handler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>Return value:</b>	ara::core::Future< ara::core::Vector< ara::core::Byte > >=0	returns a Future, which either gets readied to OperationOutput (transferResponseParameterRecord for a positive response message) or readied with ErrorCode from DiagUdsNrcErrc (for an negative response message) Data in OperationOutput.response_data will be placed after SID as transferResponseParameterRecord in the positive response.
<b>Exception Safety:</b>	noexcept	
<b>Errors:</b>	Any	Application responded with a UDS NRC value from DiagUdsNrcErrc
<b>Description:</b>	Exits an ongoing data transfer session.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.37 DataTransferReadSession class

[SWS\_DM\_01548]{DRAFT} **Definition of API class ara::diag::DataTransferReadSession** [

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

]([RS\\_Diag\\_04135](#))

#### 8.3.1.37.1 diag::DataTransferReadSession::DataTransferReadSession constructors

[SWS\_DM\_01549]{DRAFT} **Definition of API function ara::diag::DataTransferReadSession::DataTransferReadSession** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSession</a>
<b>Symbol:</b>	DataTransferReadSession()
<b>Syntax:</b>	DataTransferReadSession ()=delete;
<b>Description:</b>	No default construction allowed .

]([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01551]{DRAFT} Definition of API function ara::diag::DataTransferReadSession** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSession</a>	
<b>Symbol:</b>	DataTransferReadSession(DataTransferReadSession &&other)	
<b>Syntax:</b>	DataTransferReadSession (DataTransferReadSession &&other) noexcept=default;	
<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructs an instance of this class.	

] ([RS\\_Diag\\_04135](#))

**[SWS\_DM\_01553]{DRAFT} Definition of API function ara::diag::DataTransferReadSession** [

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

] ([RS\\_Diag\\_04135](#))

**8.3.1.37.2 diag::DataTransferReadSession::~~DataTransferReadSession de-structor**

**[SWS\_DM\_01550]{DRAFT} Definition of API function ara::diag::DataTransferReadSession::~~DataTransferReadSession** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferReadSession</a>	
<b>Symbol:</b>	~DataTransferReadSession()	
<b>Syntax:</b>	~DataTransferReadSession () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructs an instance of this class.	

] ([RS\\_Diag\\_04135](#))

### 8.3.1.37.3 diag::DataTransferReadSession::DataTransferReadSession copy assignment operator

[SWS\_DM\_01554]{DRAFT} Definition of API function ara::diag::DataTransferReadSession::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	class ara::diag::DataTransferReadSession
<b>Symbol:</b>	operator=(DataTransferReadSession const &)
<b>Syntax:</b>	auto operator= (DataTransferReadSession const &) -> DataTransferReadSession &=delete;
<b>Description:</b>	DataTransferReadSession shall be a single not assignable instance. .

](RS\_Diag\_04135)

### 8.3.1.37.4 diag::DataTransferReadSession::DataTransferReadSession move assignment operator

[SWS\_DM\_01552]{DRAFT} Definition of API function ara::diag::DataTransferReadSession::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/data_transfer.h"	
<b>Scope:</b>	class ara::diag::DataTransferReadSession	
<b>Symbol:</b>	operator=(DataTransferReadSession &&other)	
<b>Syntax:</b>	auto operator= (DataTransferReadSession &&other) & noexcept -> DataTransferReadSession & =default;	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferReadSession & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

](RS\_Diag\_04135)

### 8.3.1.38 DataTransferWriteSession class

[SWS\_DM\_01555]{DRAFT} Definition of API class ara::diag::DataTransferWriteSession [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag





<b>Symbol:</b>	DataTransferWriteSession
<b>Syntax:</b>	<code>class DataTransferWriteSession final {...};</code>
<b>Description:</b>	Encapsulates all data transfer writing variants .

]([RS\\_Diag\\_04135](#))

### 8.3.1.38.1 diag::DataTransferWriteSession::DataTransferWriteSession constructors

[SWS\_DM\_01339]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferWriteSession</a>
<b>Symbol:</b>	DataTransferWriteSession()
<b>Syntax:</b>	<code>DataTransferWriteSession ()=delete;</code>
<b>Description:</b>	No default construction allowed .

]([RS\\_Diag\\_04135](#))

[SWS\_DM\_01557]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	<a href="#">class ara::diag::DataTransferWriteSession</a>
<b>Symbol:</b>	DataTransferWriteSession(DataTransferWriteSession &&other)
<b>Syntax:</b>	<code>DataTransferWriteSession (DataTransferWriteSession &amp;&amp;other) noexcept=default;</code>
<b>Parameters (out):</b>	other      The other object
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructs an instance of this class.

]([RS\\_Diag\\_04135](#))

[SWS\_DM\_01559]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::DataTransferWriteSession [

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

]([RS\\_Diag\\_04135](#))

### 8.3.1.38.2 diag::DataTransferWriteSession::~~DataTransferWriteSession destructor

[SWS\_DM\_01556]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::~~DataTransferWriteSession [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	class ara::diag::DataTransferWriteSession
<b>Symbol:</b>	~DataTransferWriteSession()
<b>Syntax:</b>	~DataTransferWriteSession () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructs an instance of this class.

](RS\_Diag\_04135)

### 8.3.1.38.3 diag::DataTransferWriteSession::DataTransferWriteSession copy assignment operator

[SWS\_DM\_01560]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	class ara::diag::DataTransferWriteSession
<b>Symbol:</b>	operator=(DataTransferWriteSession const &)
<b>Syntax:</b>	auto operator= (DataTransferWriteSession const &) -> DataTransferWriteSession &=delete;
<b>Description:</b>	DataTransferWriteSession shall be a single not assignable instance. .

](RS\_Diag\_04135)

### 8.3.1.38.4 diag::DataTransferWriteSession::DataTransferWriteSession move assignment operator

[SWS\_DM\_01558]{DRAFT} Definition of API function ara::diag::DataTransferWriteSession::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/data_transfer.h"
<b>Scope:</b>	class ara::diag::DataTransferWriteSession
<b>Symbol:</b>	operator=(DataTransferWriteSession &&other)
<b>Syntax:</b>	auto operator= (DataTransferWriteSession &&other) & noexcept -> DataTransferWriteSession &=default;







<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	DataTransferWrite Session & =default	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns an instance of this class.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.39 ReleaseHandler class

[SWS\_DM\_01340]{DRAFT} **Definition of API class ara::diag::ReleaseHandler** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/release_handler.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	ReleaseHandler
<b>Syntax:</b>	<code>class ReleaseHandler final {...};</code>
<b>Description:</b>	ReleaseHandler contains a shared state if the processing should be canceled.

]([RS\\_Diag\\_04135](#))

#### 8.3.1.39.1 diag::ReleaseHandler::ReleaseHandler constructors

[SWS\_DM\_01530]{DRAFT} **Definition of API function ara::diag::ReleaseHandler::ReleaseHandler** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/release_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::ReleaseHandler</a>
<b>Symbol:</b>	ReleaseHandler()
<b>Syntax:</b>	<code>ReleaseHandler ()=delete;</code>
<b>Description:</b>	Default constructor of ReleaseHandler cannot be used .

]([RS\\_Diag\\_04135](#))

[SWS\_DM\_01532]{DRAFT} **Definition of API function ara::diag::ReleaseHandler::ReleaseHandler** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/release_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::ReleaseHandler</a>
<b>Symbol:</b>	ReleaseHandler(ReleaseHandler &&other)
<b>Syntax:</b>	<code>ReleaseHandler (ReleaseHandler &amp;&amp;other) noexcept;</code>





<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructs instance of class.	

](RS\_Diag\_04135)

### [SWS\_DM\_01534]{DRAFT} Definition of API function ara::diag::ReleaseHandler::ReleaseHandler [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/release_handler.h"
<b>Scope:</b>	class ara::diag::ReleaseHandler
<b>Symbol:</b>	ReleaseHandler(ReleaseHandler const &)
<b>Syntax:</b>	ReleaseHandler (ReleaseHandler const &)=delete;
<b>Description:</b>	Copy constructor of ReleaseHandler cannot be used.

](RS\_Diag\_04135)

#### 8.3.1.39.2 diag::ReleaseHandler::ReleaseHandler copy assignment operator

### [SWS\_DM\_01535]{DRAFT} Definition of API function ara::diag::ReleaseHandler::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/release_handler.h"
<b>Scope:</b>	class ara::diag::ReleaseHandler
<b>Symbol:</b>	operator=(ReleaseHandler const &)
<b>Syntax:</b>	ReleaseHandler & operator= (ReleaseHandler const &)=delete;
<b>Description:</b>	Copy assignment operator of ReleaseHandler cannot be used.

](RS\_Diag\_04135)

#### 8.3.1.39.3 diag::ReleaseHandler::ReleaseHandler move assignment operator

### [SWS\_DM\_01533]{DRAFT} Definition of API function ara::diag::ReleaseHandler::operator= [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/release_handler.h"	
<b>Scope:</b>	class ara::diag::ReleaseHandler	
<b>Symbol:</b>	operator=(ReleaseHandler &&other)	
<b>Syntax:</b>	ReleaseHandler & operator= (ReleaseHandler &&other) & noexcept;	
<b>Parameters (out):</b>	other	The other object





<b>Return value:</b>	ReleaseHandler &	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move assigns instance of class.	

]([RS\\_Diag\\_04135](#))

#### 8.3.1.39.4 diag::ReleaseHandler::~~ReleaseHandler destructor

[SWS\_DM\_01531]{DRAFT} Definition of API function `ara::diag::ReleaseHandler::~~ReleaseHandler` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/release_handler.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ReleaseHandler</a>	
<b>Symbol:</b>	~ReleaseHandler()	
<b>Syntax:</b>	~ReleaseHandler () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Default destructor.	

]([RS\\_Diag\\_04135](#))

#### 8.3.1.39.5 diag::ReleaseHandler::MayRelease function

[SWS\_DM\_01536]{DRAFT} Definition of API function `ara::diag::ReleaseHandler::MayRelease` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/release_handler.h"	
<b>Scope:</b>	<a href="#">class ara::diag::ReleaseHandler</a>	
<b>Symbol:</b>	MayRelease()	
<b>Syntax:</b>	bool MayRelease () const noexcept;	
<b>Return value:</b>	bool	True in if the shared resource is no longer in use, False otherwise
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Reports whether the shared resource is no longer in use.	

]([RS\\_Diag\\_04135](#))

### 8.3.1.39.6 diag::ReleaseHandler::SetNotifier function

[SWS\_DM\_01537]{DRAFT} **Definition of API function ara::diag::ReleaseHandler::SetNotifier** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/release_handler.h"	
<b>Scope:</b>	class ara::diag::ReleaseHandler	
<b>Symbol:</b>	SetNotifier(std::function< void()> notifier)	
<b>Syntax:</b>	void SetNotifier (std::function< void()> notifier);	
<b>Parameters (in):</b>	notifier	Notification function that is called upon releasing the shared resource
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

](RS\_Diag\_04135)

## 8.3.2 C++ Diagnostic generated API Interfaces

Namespaces are used to separate the definition of services from each other to prevent name conflicts and they allow to use reasonably short names.

[SWS\_DM\_00510]{DRAFT} **Namespace of Service header files** [Based on the [symbol](#) attributes of the ordered [SymbolProps](#) aggregated by [PortInterface](#) in role [namespace](#), the C++ namespace of the *Service header file* shall be:

```

1 namespace <PortInterface.namespace[0].symbol> {
2 namespace <PortInterface.namespace[1].symbol> {
3 namespace <...> {
4 namespace <PortInterface.namespace[n].symbol> {
5 ...
6 } // namespace <PortInterface.namespace[n].symbol>
7 } // namespace <...>
8 } // namespace <PortInterface.namespace[1].symbol>
9 } // namespace <PortInterface.namespace[0].symbol>
    
```

with all namespace names converted to lower-case letters.](RS\_AP\_00114)

### 8.3.2.1 Implementation Types header files

The *Implementation Types header files* include the ara::diag specific type declarations derived from the [CppImplementationDataTypes](#) created from the definitions of AUTOSAR meta model classes within the [DiagnosticPortInterface](#) description.

[SWS\_DM\_00511]{DRAFT} **Implementation Types header files existence** [The diagnostic management shall provide an *Implementation Types header file* for each [CppImplementationDataType](#) defined in the input by using the file name

impl\_type\_<symbol>.h, where <symbol> is the *Cpp Implementation Data Type symbol* converted to lower-case letters.](RS\_AP\_00116)

The *Implementation Types header files* might need to include other header files, e.g. for `ara::core::String` or `ara::core::Vector`.

**[SWS\_DM\_00512]{DRAFT} Data Type definitions for AUTOSAR Data Types in Implementation Types header files** [The *Implementation Types header files* shall include all the used definitions for types, structures and classes of the AUTOSAR Data Types.](RS\_AP\_00114)

**[SWS\_DM\_00513]{DRAFT} Implementation Types header file namespace** [The C++ namespace of the *Implementation Types header file* for a given `CppImplementationDataType` is defined via the aggregated `namespace`. Based on the `symbol` attributes of the ordered `SymbolProps` aggregated by `CppImplementationDataType` in role `namespace`, the C++ namespace of the *Implementation Types header file* shall be:

```

1 namespace <CppImplementationDataType.namespace[0].symbol> {
2 namespace <CppImplementationDataType.namespace[1].symbol> {
3 namespace <...> {
4 namespace <CppImplementationDataType.namespace[n].symbol> {
5 ...
6 } // namespace <CppImplementationDataType.namespace[n].symbol>
7 } // namespace <...>
8 } // namespace <CppImplementationDataType.namespace[1].symbol>
9 } // namespace <CppImplementationDataType.namespace[0].symbol>
    
```

with all namespace names converted to lower-case letters.](RS\_AP\_00114)

### 8.3.2.2 Typed Routine class

The `ara::core::InstanceSpecifier` is only compatible with `PortInterface` of `DiagnosticRoutineInterface`.

**[SWS\_DM\_00604]{DRAFT} Definition of API class `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface`** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
<b>Symbol:</b>	ShortnameOfRIPortInterface
<b>Syntax:</b>	class ShortnameOfRIPortInterface {...};
<b>Description:</b>	Typed Routine interface.

](RS\_Main\_01002, RS\_Diag\_04169)

### 8.3.2.2.1 Routine::StartOutput type

**[SWS\_DM\_00581]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::StartOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
<b>Symbol:</b>	StartOutput
<b>Syntax:</b>	struct StartOutput {...};
<b>Description:</b>	Response data.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.2.2.2 Routine::StopOutput type

**[SWS\_DM\_00582]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::StopOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
<b>Symbol:</b>	StopOutput
<b>Syntax:</b>	struct StopOutput {...};
<b>Description:</b>	Response data.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.2.2.3 Routine::RequestResultsOutput type

**[SWS\_DM\_00583]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::RequestResultsOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
<b>Symbol:</b>	RequestResultsOutput
<b>Syntax:</b>	struct RequestResultsOutput {...};
<b>Description:</b>	Response data.

]([RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.2.2.4 Routine constructor

**[SWS\_DM\_00589]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::ShortnameOfRIPortInterface** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_routine.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>	
<b>Symbol:</b>	ShortnameOfRIPortInterface(const ara::core::InstanceSpecifier &specifier, ara::diag::ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit ShortnameOfRIPortInterface (const ara::core::InstanceSpecifier &specifier, <a href="#">ara::diag::ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of typed Routine interface.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#))

### 8.3.2.2.5 Routine move constructor

**[SWS\_DM\_01670]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::ShortnameOfRIPortInterface** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_routine.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>	
<b>Symbol:</b>	ShortnameOfRIPortInterface(ShortnameOfRIPortInterface &&)	
<b>Syntax:</b>	ShortnameOfRIPortInterface (ShortnameOfRIPortInterface &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of ShortnameOfRIPortInterface.	

]([RS\\_AP\\_00147](#))

### 8.3.2.2.6 Routine copy constructor

**[SWS\_DM\_01669]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::ShortnameOfRIPortInterface** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>
<b>Symbol:</b>	ShortnameOfRIPortInterface(ShortnameOfRIPortInterface &)
<b>Syntax:</b>	ShortnameOfRIPortInterface (ShortnameOfRIPortInterface &)=delete;
<b>Description:</b>	ShortnameOfRIPortInterface shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### 8.3.2.2.7 Routine move operator

**[SWS\_DM\_01668]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::operator=** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>
<b>Symbol:</b>	operator=(ShortnameOfRIPortInterface &&)
<b>Syntax:</b>	<a href="#">ShortnameOfRIPortInterface</a> & operator= ( <a href="#">ShortnameOfRIPortInterface</a> &&)=delete;
<b>Description:</b>	Move assignment operator of ShortnameOfRIPortInterface.

]([RS\\_AP\\_00147](#))

### 8.3.2.2.8 Routine copy operator

**[SWS\_DM\_01667]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::operator=** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>
<b>Symbol:</b>	operator=(ShortnameOfRIPortInterface &)
<b>Syntax:</b>	<a href="#">ShortnameOfRIPortInterface</a> & operator= ( <a href="#">ShortnameOfRIPortInterface</a> &)=delete;
<b>Description:</b>	ShortnameOfRIPortInterface shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))



### 8.3.2.2.9 Routine destructor

**[SWS\_DM\_00590]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::~~ShortnameOfRIPortInterface** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
<b>Symbol:</b>	~ShortnameOfRIPortInterface()
<b>Syntax:</b>	virtual ~ShortnameOfRIPortInterface () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of typed Routine interface.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04169](#))

### 8.3.2.2.10 Routine::Offer function

**[SWS\_DM\_00594]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Offer** [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/name_routine.h"		
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface		
<b>Symbol:</b>	Offer()		
<b>Syntax:</b>	ara::core::Result< void > Offer ();		
<b>Return value:</b>	ara::core::Result< void >   -		
<b>Exception Safety:</b>	not exception safe		
<b>Errors:</b>	<table border="1"> <tr> <td>DiagErrorDomain::DiagOfferErrc::kAlreadyOffered</td> <td>This service was already offered.</td> </tr> </table>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.		
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.		

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04224](#))

### 8.3.2.2.11 Routine::StopOffer function

**[SWS\_DM\_00595]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::StopOffer** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_routine.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();





<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Main\\_01002](#), [RS\\_Diag\\_04169](#))

### 8.3.2.2.12 Routine::Start function

[SWS\_DM\_00591]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Start [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_routine.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface</a>	
<b>Symbol:</b>	Start(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype,..., const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">StartOutput</a> > Start (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype,..., const <a href="#">ara::diag::MetaInfo</a> &metaInfo, <a href="#">ara::diag::CancellationHandler</a> cancellationHandler)=0;	
<b>Parameters (in):</b>	shortname1OfArgumentDataPrototype	Content of DataElement#1 in response message
	shortname2OfArgumentDataPrototype	Content of DataElement#2 in response message
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>DIRECTION NOT DEFINED</b>	...	-
<b>Return value:</b>	ara::core::Future< StartOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction Start request for this DiagnosticRoutineIdentifier.	

[|\(RS\\_AP\\_00119, RS\\_AP\\_00138, RS\\_Diag\\_04169, RS\\_Diag\\_04170, RS\\_Diag\\_04224\)](#)

### 8.3.2.2.13 Routine::Stop function

#### [SWS\_DM\_00592]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Stop [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_routine.h"	
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
<b>Symbol:</b>	Stop(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype, ..., const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< StopOutput > Stop (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype, ..., const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)=0;	
<b>Parameters (in):</b>	shortname1OfArgumentDataPrototype	Content of DataElement#1 in response message
	shortname2OfArgumentDataPrototype	Content of DataElement#2 in response message
	metaInfo	contains additional meta information





	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>DIRECTION NOT DEFINED</b>	...	-
<b>Return value:</b>	ara::core::Future< Stop Output >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.



△

	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction Stop request for this DiagnosticRoutineIdentifier.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04224)



### 8.3.2.2.14 Routine::RequestResults function

#### [SWS\_DM\_00593]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::RequestResults [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_routine.h"	
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
<b>Symbol:</b>	RequestResults(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype, ..., const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; RequestResultsOutput &gt; RequestResults (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfTypeOfArgumentData Prototype::Type2OfArgumentDataPrototype shortname2OfArgumentData Prototype, ..., const ara::diag::MetaInfo &amp;metaInfo, ara::diag::CancellationHandler cancellationHandler)=0;</pre>	
<b>Parameters (in):</b>	shortname1OfArgumentDataPrototype	Content of DataElement#1 in response message
	shortname2OfArgumentDataPrototype	Content of DataElement#2 in response message
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>DIRECTION NOT DEFINED</b>	...	-
<b>Return value:</b>	ara::core::Future< RequestResultsOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for RoutineControl with SubFunction RequestResults request for this DiagnosticRoutine Identifier.	

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170, RS\_Diag\_04224)

### 8.3.2.3 Typed DataIdentifier class

The *InstanceSpecifier* is only compatible with `PortInterface` of `DiagnosticDataIdentifierInterface`.

**[SWS\_DM\_00601]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
<b>Symbol:</b>	ShortnameOfDIPortInterface
<b>Syntax:</b>	class ShortnameOfDIPortInterface {...};
<b>Description:</b>	Typed DataIdentifier interface.

|(RS\_Diag\_04169)

### 8.3.2.3.1 DataIdentifier::OperationOutput type

[SWS\_DM\_00579]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Output [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
<b>Symbol:</b>	Output
<b>Syntax:</b>	struct Output {...};
<b>Description:</b>	Response data.

]([RS\\_Diag\\_04169](#))

### 8.3.2.3.2 DataIdentifier constructor

[SWS\_DM\_00585]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"	
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface	
<b>Symbol:</b>	ShortnameOfDIPortInterface(const ara::core::InstanceSpecifier &specifier, ara::diag::DataIdentifierReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit ShortnameOfDIPortInterface (const ara::core::InstanceSpecifier &specifier, <a href="#">ara::diag::DataIdentifierReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant for reads, writes or both
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of typed DataIdentifier interface.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#))

### 8.3.2.3.3 DataIdentifier move constructor

[SWS\_DM\_01666]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
<b>Symbol:</b>	ShortnameOfDIPortInterface(ShortnameOfDIPortInterface &&)
<b>Syntax:</b>	ShortnameOfDIPortInterface (ShortnameOfDIPortInterface &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of ShortnameOfDIPortInterface.

](RS\_AP\_00147)

### 8.3.2.3.4 DataIdentifier copy constructor

[SWS\_DM\_01665]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
<b>Symbol:</b>	ShortnameOfDIPortInterface(ShortnameOfDIPortInterface &)
<b>Syntax:</b>	ShortnameOfDIPortInterface (ShortnameOfDIPortInterface &)=delete;
<b>Description:</b>	ShortnameOfDIPortInterface shall be a single not copy-able instance.

](RS\_AP\_00147)

### 8.3.2.3.5 DataIdentifier move operator

[SWS\_DM\_01664]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
<b>Symbol:</b>	operator=(ShortnameOfDIPortInterface &&)
<b>Syntax:</b>	ShortnameOfDIPortInterface & operator= (ShortnameOfDIPortInterface &&)=delete;
<b>Description:</b>	Move assignment operator of ShortnameOfDIPortInterface.

](RS\_AP\_00147)

### 8.3.2.3.6 DataIdentifier copy operator

[SWS\_DM\_01663]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>
<b>Symbol:</b>	operator=(ShortnameOfDIPortInterface &)
<b>Syntax:</b>	<a href="#">ShortnameOfDIPortInterface</a> & operator= ( <a href="#">ShortnameOfDIPortInterface</a> &)=delete;
<b>Description:</b>	ShortnameOfDIPortInterface shall be a single not assignable instance.

]([RS\\_AP\\_00147](#))

### 8.3.2.3.7 DataIdentifier destructor

[SWS\_DM\_00586]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::~~ShortnameOfDIPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>
<b>Symbol:</b>	~ShortnameOfDIPortInterface()
<b>Syntax:</b>	virtual ~ShortnameOfDIPortInterface () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of typed DataIdentifier interface.

]([RS\\_AP\\_00134](#), [RS\\_Diag\\_04169](#))

### 8.3.2.3.8 DataIdentifier::Offer function

[SWS\_DM\_00599]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Offer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>
<b>Symbol:</b>	Offer()
<b>Syntax:</b>	ara::core::Result< void > Offer ();
<b>Return value:</b>	ara::core::Result< void >   -
<b>Exception Safety:</b>	not exception safe





<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.2.3.9 DataIdentifier::StopOffer function

[SWS\_DM\_00600]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04169](#))

### 8.3.2.3.10 DataIdentifier::Read function

[SWS\_DM\_00640]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Read [

<b>Kind:</b>	function				
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"				
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>				
<b>Symbol:</b>	Read(const ara::diag::MetaInfo &metaInfo, ara::diag::CancellationHandler cancellationHandler)				
<b>Syntax:</b>	virtual ara::core::Future< Output > Read (const <a href="#">ara::diag::MetaInfo</a> &metaInfo, <a href="#">ara::diag::CancellationHandler</a> cancellationHandler)=0;				
<b>Parameters (in):</b>	<table border="1"> <tr> <td>metaInfo</td> <td>contains additional meta information</td> </tr> <tr> <td>cancellationHandler</td> <td>This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).</td> </tr> </table>	metaInfo	contains additional meta information	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
metaInfo	contains additional meta information				
cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).				
<b>Return value:</b>	<table border="1"> <tr> <td>ara::core::Future&lt; Output &gt;</td> <td>a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)</td> </tr> </table>	ara::core::Future< Output >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)		
ara::core::Future< Output >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)				
<b>Exception Safety:</b>	not exception safe				







<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.





	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for ReadDataByIdentifier request for this DiagnosticDataIdentifier.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00138](#), [RS\\_Diag\\_04169](#), [RS\\_Diag\\_04170](#))

### 8.3.2.3.11 DataIdentifier::Write function

[SWS\_DM\_00598]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface::Write [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_identifier.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface</a>
<b>Symbol:</b>	Write(Namespace1OfArgumentDataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentDataPrototype, Namespace2OfArgumentDataPrototype::Type2OfArgumentDataPrototype shortname2OfArgumentDataPrototype, ..., const ara::diag::MetaInfo &metaInfo, CancellationHandler cancellationHandler)





<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; void &gt; Write (Namespace1OfTypeOfArgument DataPrototype::Type1OfArgumentDataPrototype shortname1OfArgumentData Prototype, Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgument DataPrototype shortname2OfArgumentDataPrototype, ..., const ara::diag::MetaInfo &amp;metaInfo, CancellationHandler cancellation Handler)=0;</pre>	
<b>Parameters (in):</b>	shortname1OfArgument DataPrototype	Content of DataElement#1 in request message
	shortname2OfArgument DataPrototype	Content of DataElement#2 in request message
	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling cancellationHandler.IsCanceled() (which returns true in case the current conversation has been cancelled and false otherwise), -Register a notifier function by calling cancellationHandler.SetNotifier() (the registered notifier is called if the current conversation is cancelled).
<b>DIRECTION NOT DEFINED</b>	...	-
<b>Return value:</b>	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent	0x25, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action	0x26, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range	0x31, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginesRunning	0x83, Negative return code according to ISO 14229.





DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
<b>Description:</b>	Called for WriteDataByIdentifier request for this DiagnosticDataIdentifier.

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04169, RS\_Diag\_04170)

### 8.3.2.4 Typed DataElement class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [Diagnostic-DataElementInterface](#).

**[SWS\_DM\_00603]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
<b>Symbol:</b>	ShortnameOfDEPortInterface
<b>Syntax:</b>	class ShortnameOfDEPortInterface {...};
<b>Description:</b>	Typed DataElement interface.

]([RS\\_Diag\\_04169](#))

#### 8.3.2.4.1 DataElement::OperationOutput type

**[SWS\_DM\_00580]{DRAFT} Definition of API class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::OperationOutput** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>
<b>Symbol:</b>	OperationOutput
<b>Syntax:</b>	struct OperationOutput {...};
<b>Description:</b>	Response data.

]([RS\\_Diag\\_04169](#))

#### 8.3.2.4.2 DataElement constructor

**[SWS\_DM\_00587]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>





<b>Symbol:</b>	ShortnameOfDEPortInterface(const ara::core::InstanceSpecifier &specifier, ara::diag::ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit ShortnameOfDEPortInterface (const ara::core::InstanceSpecifier &specifier, <a href="#">ara::diag::ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of typed DataElement interface.	

]([RS\\_AP\\_00137](#), [RS\\_Diag\\_04169](#))

### 8.3.2.4.3 DataElement move constructor

[SWS\_DM\_01662][DRAFT] Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>
<b>Symbol:</b>	ShortnameOfDEPortInterface(ShortnameOfDEPortInterface &&)
<b>Syntax:</b>	ShortnameOfDEPortInterface (ShortnameOfDEPortInterface &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of ShortnameOfDEPortInterface.

]([RS\\_AP\\_00147](#))

### 8.3.2.4.4 DataElement copy constructor

[SWS\_DM\_01661][DRAFT] Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>
<b>Symbol:</b>	ShortnameOfDEPortInterface(ShortnameOfDEPortInterface &)
<b>Syntax:</b>	ShortnameOfDEPortInterface (ShortnameOfDEPortInterface &)=delete;
<b>Description:</b>	ShortnameOfDEPortInterface shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))



### 8.3.2.4.5 DataElement move operator

[SWS\_DM\_01660]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface
<b>Symbol:</b>	operator=(ShortnameOfDEPortInterface &&)
<b>Syntax:</b>	ShortnameOfDEPortInterface & operator= (ShortnameOfDEPortInterface &&)=delete;
<b>Description:</b>	Move assignment operator of ShortnameOfDEPortInterface.

](RS\_AP\_00147)

### 8.3.2.4.6 DataElement copy operator

[SWS\_DM\_01659]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface
<b>Symbol:</b>	operator=(ShortnameOfDEPortInterface &)
<b>Syntax:</b>	ShortnameOfDEPortInterface & operator= (ShortnameOfDEPortInterface &)=delete;
<b>Description:</b>	ShortnameOfDEPortInterface shall be a single not assignable instance.

](RS\_AP\_00147)

### 8.3.2.4.7 DataElement destructor

[SWS\_DM\_00588]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::~~ShortnameOfDEPortInterface [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/name_data_element.h"
<b>Scope:</b>	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface
<b>Symbol:</b>	~ShortnameOfDEPortInterface()
<b>Syntax:</b>	virtual ~ShortnameOfDEPortInterface () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of typed DataElement interface.

](RS\_AP\_00134, RS\_Diag\_04169)

### 8.3.2.4.8 DataElement::Offer function

[SWS\_DM\_00597]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_data_element.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [RS\\_Diag\\_04169](#))

### 8.3.2.4.9 DataElement::StopOffer function

[SWS\_DM\_00617]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_data_element.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	void StopOffer ();	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04169](#))

### 8.3.2.4.10 DataElement::Read function

[SWS\_DM\_00596]{DRAFT} Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Read [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/name_data_element.h"	
<b>Scope:</b>	<a href="#">class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface</a>	
<b>Symbol:</b>	Read(const ara::diag::MetalInfo &metalInfo, ara::diag::CancellationHandler cancellationHandler)	





<b>Syntax:</b>	<code>virtual ara::core::Future&lt; <a href="#">OperationOutput</a> &gt; Read (const <a href="#">ara::diag::MetaInfo</a> &amp;metaInfo, <a href="#">ara::diag::CancellationHandler</a> cancellationHandler)=0;</code>	
<b>Parameters (in):</b>	metaInfo	contains additional meta information
	cancellationHandler	This parameter used to -Query the current cancellation status by calling <code>cancellationHandler.IsCanceled()</code> (which 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).
<b>Return value:</b>	<code>ara::core::Future&lt; <a href="#">OperationOutput</a> &gt;</code>	a Result with either <code>OperationOutput</code> (for a positive response message) or an UDS NRC value (for an negative response message)
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneralReject</code>	0x10, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot Supported</code>	0x11, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubfunctionNot Supported</code>	0x12, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kIncorrectMessage LengthOrInvalidFormat</code>	0x13, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kResponseToo Long</code>	0x14, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kBusyRepeat Request</code>	0x21, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kConditionsNot Correct</code>	0x22, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestSequence Error</code>	0x24, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kNoResponseFrom SubnetComponent</code>	0x25, Negative return code according to ISO 14229.
	<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kFailurePrevents ExecutionOfRequested Action</code>	0x26, Negative return code according to ISO 14229.
<code>DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequestOutOf Range</code>	0x31, Negative return code according to ISO 14229.	





DiagUdsNrcError Domain::DiagUdsNrc Errc::kSecurityAccess Denied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kExceedNumberOf Attempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRequiredTime DelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kUploadDownload NotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransferData Suspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kGeneral ProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kWrongBlock SequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kSubFunctionNot SupportedInActive Session	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kServiceNot SupportedInActive Session	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEnginelsNot Running	0x84, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kEngineRunTime TooLow	0x85, Negative return code according to ISO 14229.



△

DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo High	0x86, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTemperatureToo Low	0x87, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo High	0x88, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVehicleSpeedToo Low	0x89, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo High	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kThrottlePedalToo Low	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTransmission RangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kBrakeSwitchNot Closed	0x8F, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kShifterLeverNotIn Park	0x90, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kTorqueConverter ClutchLocked	0x91, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
DiagUdsNrcError Domain::DiagUdsNrc Errc::kResource TemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
<b>Description:</b>	Called for reading a DataElement.

|(RS\_AP\_00119, RS\_AP\_00138, RS\_Diag\_04170, RS\_Diag\_04169)

### 8.3.3 C++ Diagnostic SOVD API Interfaces

#### 8.3.3.1 SOVD Authorization

##### 8.3.3.1.1 SovdAuthorization class

[SWS\_DM\_01490]{DRAFT} Definition of API class `ara::diag::SovdAuthorization` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>SovdAuthorization</code>
<b>Syntax:</b>	<code>class SovdAuthorization {...};</code>
<b>Description:</b>	Class to implement the SOVD Authorization interfaces in the application.

]([RS\\_Diag\\_04268](#))

##### 8.3.3.1.2 SovdAuthorization::SovdAuthorization constructor

[SWS\_DM\_01489]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::SovdAuthorization` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdAuthorization</code></a>	
<b>Symbol:</b>	<code>SovdAuthorization(ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType)</code>	
<b>Syntax:</b>	<code>explicit SovdAuthorization (ara::core::InstanceSpecifier instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;</code>	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdAuthorization service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructs the port for SOVD Authorization interface.	

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.3 SovdAuthorization::SovdAuthorization destructor

[SWS\_DM\_01488]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::~~SovdAuthorization` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>
<b>Symbol:</b>	~SovdAuthorization()
<b>Syntax:</b>	virtual ~SovdAuthorization () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SovdAuthorization.

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.4 SovdAuthorization::GetTokenUrl function

[SWS\_DM\_01484]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::GetTokenUrl` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>	
<b>Symbol:</b>	GetTokenUrl(const ara::core::Map< const ara::core::StringView, const ara::core::Vector< const ara::core::StringView > > &queryParams, const MetaInfo &metaInfo)	
<b>Syntax:</b>	virtual ara::core::Future< ara::core::String > GetTokenUrl (const ara::core::Map< const ara::core::StringView, const ara::core::Vector< const ara::core::StringView > > &queryParams, const <a href="#">MetaInfo</a> &metaInfo)=0;	
<b>Parameters (in):</b>	queryParams	The query parameters passed in the request.
	metaInfo	The meta information of a diagnostic service port.
<b>Return value:</b>	ara::core::Future< ara::core::String >	The URL that the SOVD client shall use to request a token.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.5 SovdAuthorization::GetAuthorizationUrl function

[SWS\_DM\_01485]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::GetAuthorizationUrl` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>	
<b>Symbol:</b>	GetAuthorizationUrl(const ara::core::Map< const ara::core::StringView, const ara::core::Vector< const ara::core::StringView > > &queryParameters, const MetaInfo &metaInfo)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; ara::core::String &gt; GetAuthorizationUrl (const ara::core::Map&lt; const ara::core::StringView, const ara::core::Vector&lt; const ara::core::StringView &gt; &gt; &amp;queryParameters, const MetaInfo &amp;metaInfo)=0;</pre>	
<b>Parameters (in):</b>	queryParameters	The query parameters passed in the request.
	metaInfo	The meta information of a diagnostic service port.
<b>Return value:</b>	ara::core::Future< ara::core::String >	The URL that the SOVD client shall use for authorization.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	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.	

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.6 SovdAuthorization::Offer function

[SWS\_DM\_01487]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
	DiagErrorDomain::Diag OfferErrc::kConfiguration Mismatch	Implementation does not fit to the configuration.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_Diag\\_04268](#))



### 8.3.3.1.7 SovdAuthorization::StopOffer function

[SWS\_DM\_01486]{DRAFT} Definition of API function `ara::diag::SovdAuthorization::StopOffer` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	<code>void StopOffer ();</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.8 SovdAuthorization::ValidateAuthorizationOutput type

[SWS\_DM\_01927]{DRAFT} Definition of API class `ara::diag::SovdAuthorization::ValidateAuthorizationOutput` [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdAuthorization</a>
<b>Symbol:</b>	ValidateAuthorizationOutput
<b>Syntax:</b>	<code>struct ValidateAuthorizationOutput {...};</code>
<b>Description:</b>	Return type for ValidateAuthorization().

]([RS\\_Diag\\_04268](#))

### 8.3.3.1.9 SovdAuthorization::ValidateAuthorizationOutput identity

[SWS\_DM\_01819]{DRAFT} Definition of API variable `ara::diag::SovdAuthorization::ValidateAuthorizationOutput::identity` [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput</a>
<b>Symbol:</b>	identity
<b>Type:</b>	<code>ara::core::Optional&lt; ara::core::String &gt;</code>
<b>Syntax:</b>	<code>ara::core::Optional&lt;ara::core::String&gt; identity;</code>





<b>Description:</b>	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.
---------------------	---

](RS\_Diag\_04268)

### 8.3.3.1.10 SovdAuthorization::ValidateAuthorizationOutput valid until

[SWS\_DM\_01820]{DRAFT} Definition of API variable ara::diag::SovdAuthorization::ValidateAuthorizationOutput::validUntil [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"
<b>Scope:</b>	struct ara::diag::SovdAuthorization::ValidateAuthorizationOutput
<b>Symbol:</b>	validUntil
<b>Type:</b>	std::chrono::system_clock::time_point
<b>Syntax:</b>	std::chrono::system_clock::time_point validUntil;
<b>Description:</b>	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).

](RS\_Diag\_04268)

### 8.3.3.1.11 SovdAuthorization::ValidateAuthorization function

[SWS\_DM\_01483]{DRAFT} Definition of API function ara::diag::SovdAuthorization::ValidateAuthorization [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_authorization.h"	
<b>Scope:</b>	class ara::diag::SovdAuthorization	
<b>Symbol:</b>	ValidateAuthorization(ara::core::StringView token, MetaInfo &metaInfo, ClientAuthentication &clientAuthentication)	
<b>Syntax:</b>	virtual ara::core::Future< ValidateAuthorizationOutput > ValidateAuthorization (ara::core::StringView token, MetaInfo &metaInfo, ClientAuthentication &clientAuthentication)=0;	
<b>Parameters (in):</b>	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.





<b>Return value:</b>	ara::core::Future< ValidateAuthorization Output >	Either ValidateAuthorizationOutput (for a successful validation) or an error code.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagSovdErrc::kInvalid	AccessToken, In case the access token is invalid.
	DiagSovdErrc::kTemporarilyNotAvailable,	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).
<b>Description:</b>	This function validates the authorization token sent by the client.  The application shall validate the authorization token according to the authorization protocol used in the system and then according to the outcome set the respective roles as authorized in the clientAuthentication that is passed as parameter. If OAuth 2.0 is used, this would implement the validation that is supposed to happen in the Resource Server.	

|(RS\_Diag\_04268)

### 8.3.3.2 SOVD Proximity Challenge

#### 8.3.3.2.1 SovdProximityChallenge class

[SWS\_DM\_01481]{DRAFT} Definition of API class ara::diag::SovdProximityChallenge [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SovdProximityChallenge
<b>Syntax:</b>	class SovdProximityChallenge {...};
<b>Description:</b>	Class to implement the SOVD Proximity Challenge interfaces in the application.

|(RS\_Diag\_04262)

#### 8.3.3.2.2 SovdProximityChallenge::SovdProximityChallengeType type

[SWS\_DM\_01482]{DRAFT} Definition of API class ara::diag::SovdProximityChallengeType [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SovdProximityChallengeType
<b>Syntax:</b>	struct SovdProximityChallengeType {...};
<b>Description:</b>	Holds the data returned by the SovdProximityChallenge::GetChallenge()

|(RS\_Diag\_04262)

### 8.3.3.2.3 SovdProximityChallenge::SovdProximityChallengeType valid until

[SWS\_DM\_01491]{DRAFT} Definition of API variable `ara::diag::SovdProximityChallengeType::valid_until` [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdProximityChallengeType</a>
<b>Symbol:</b>	valid_until
<b>Type:</b>	std::chrono::system_clock::time_point
<b>Syntax:</b>	std::chrono::system_clock::time_point valid_until;
<b>Description:</b>	The time until the challenge is valid.

] ([RS\\_Diag\\_04262](#))

### 8.3.3.2.4 SovdProximityChallenge::SovdProximityChallengeType challenge

[SWS\_DM\_01492]{DRAFT} Definition of API variable `ara::diag::SovdProximityChallengeType::challenge` [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdProximityChallengeType</a>
<b>Symbol:</b>	challenge
<b>Type:</b>	ara::core::Span< ara::core::Byte >
<b>Syntax:</b>	ara::core::Span<ara::core::Byte> challenge;
<b>Description:</b>	Challenge which has to be solved by the SOVD client to prove proximity.

] ([RS\\_Diag\\_04262](#))

### 8.3.3.2.5 SovdProximityChallenge::SovdProximityChallenge constructor

[SWS\_DM\_01480]{DRAFT} Definition of API function `ara::diag::SovdProximityChallenge::SovdProximityChallenge` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdProximityChallenge</a>	
<b>Symbol:</b>	SovdProximityChallenge(ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit SovdProximityChallenge (ara::core::InstanceSpecifier instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdProximityChallenge service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant





<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Constructs the port for SOVD Proximity Challenge interface.

](RS\_Diag\_04262)

### 8.3.3.2.6 SovdProximityChallenge::SovdProximityChallenge destructor

[SWS\_DM\_01479]{DRAFT} Definition of API function `ara::diag::SovdProximityChallenge::~~SovdProximityChallenge` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdProximityChallenge</a>
<b>Symbol:</b>	~SovdProximityChallenge()
<b>Syntax:</b>	<code>virtual ~SovdProximityChallenge () noexcept=default;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SovdProximityChallenge.

](RS\_Diag\_04262)

### 8.3.3.2.7 SovdProximityChallenge::GetChallenge function

[SWS\_DM\_01494]{DRAFT} Definition of API function `ara::diag::SovdProximityChallenge::GetChallenge` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdProximityChallenge</a>	
<b>Symbol:</b>	GetChallenge(const MetaInfo &metaInfo)	
<b>Syntax:</b>	<code>virtual ara::core::Future&lt; SovdProximityChallengeType &gt; GetChallenge (const MetaInfo &amp;metaInfo)=0;</code>	
<b>Parameters (in):</b>	metaInfo	The meta information of a diagnostic service port
<b>Return value:</b>	ara::core::Future< SovdProximityChallengeType >	Challenge created by the application and point in time until when the challenge is valid (valid_until)
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This function returns a proximity challenge for the client.	

](RS\_Diag\_04262)

### 8.3.3.2.8 SovdProximityChallenge::Offer function

[SWS\_DM\_01478]{DRAFT} Definition of API function `ara::diag::SovdProximityChallenge::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdProximityChallenge</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; Offer ();</code>	
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
	DiagErrorDomain::Diag OfferErrc::kConfiguration Mismatch	Implementation does not fit to the configuration.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_Diag\\_04262](#))

### 8.3.3.2.9 SovdProximityChallenge::StopOffer function

[SWS\_DM\_01495]{DRAFT} Definition of API function `ara::diag::SovdProximityChallenge::StopOffer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdProximityChallenge</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	<code>void StopOffer ();</code>	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04262](#))

### 8.3.3.2.10 SovdProximityChallenge::ValidateResponse function

#### [SWS\_DM\_01493]{DRAFT} Definition of API function ara::diag::SovdProximityChallenge::ValidateResponse [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_proximity_challenge.h"	
<b>Scope:</b>	class ara::diag::SovdProximityChallenge	
<b>Symbol:</b>	ValidateResponse(ara::core::Span< ara::core::Byte > proximityResponse, MetaInfo &metaInfo)	
<b>Syntax:</b>	virtual ara::core::Future< void > ValidateResponse (ara::core::Span< ara::core::Byte > proximityResponse, MetaInfo &metaInfo)=0;	
<b>Parameters (in):</b>	proximityResponse	The proximity_response sent by the client
	metaInfo	The meta information of a diagnostic service port
<b>Return value:</b>	ara::core::Future< void >	Value if the response was valid
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This function validates the proximity response sent by the client.	

]([RS\\_Diag\\_04262](#))

### 8.3.3.3 SOVD Service Validation

#### 8.3.3.3.1 diag::SovdRequestMethod

#### [SWS\_DM\_01821]{DRAFT} Definition of API enum ara::diag::SovdRequestMethod [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	SovdRequestMethod	
<b>Underlying type:</b>	-	
<b>Syntax:</b>	enum class SovdRequestMethod {...};	
<b>Values:</b>	kGet	-
	kPut	< GET.PUT.
	kPost	POST.
	kDelete	DELETE.
<b>Description:</b>	HTTP method used in the SOVD request.	

]([RS\\_Diag\\_04256](#))

### 8.3.3.3.2 SovdServiceValidation class

[SWS\_DM\_01822]{DRAFT} Definition of API class `ara::diag::SovdServiceValidation` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>SovdServiceValidation</code>
<b>Syntax:</b>	<code>class SovdServiceValidation {...};</code>
<b>Description:</b>	Class to implement manufacturer-/supplier-specific checks for SOVD in the application.

]([RS\\_Diag\\_04256](#))

### 8.3.3.3.3 SovdServiceValidation::SovdServiceValidation constructor

[SWS\_DM\_01823]{DRAFT} Definition of API function `ara::diag::SovdServiceValidation::SovdServiceValidation` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"	
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdServiceValidation</code></a>	
<b>Symbol:</b>	<code>SovdServiceValidation(ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType)</code>	
<b>Syntax:</b>	<code>explicit SovdServiceValidation (ara::core::InstanceSpecifier instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;</code>	
<b>Parameters (in):</b>	<code>instanceSpecifier</code>	InstanceSpecifier to a PortPrototype of a <code>SovdServiceValidation</code> service instance in the manifest
	<code>reentrancyType</code>	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	<code>noexcept</code>	
<b>Description:</b>	Constructs the port for <code>SovdServiceValidation</code> interface.	

]([RS\\_Diag\\_04256](#))

### 8.3.3.3.4 SovdServiceValidation::~SovdServiceValidation destructor

[SWS\_DM\_01824]{DRAFT} Definition of API function `ara::diag::SovdServiceValidation::~~SovdServiceValidation` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdServiceValidation</code></a>
<b>Symbol:</b>	<code>~SovdServiceValidation()</code>
<b>Syntax:</b>	<code>virtual ~SovdServiceValidation () noexcept=default;</code>







<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SovdServiceValidation.

](RS\_Diag\_04256)

### 8.3.3.3.5 SovdServiceValidation::Offer function

[SWS\_DM\_01825][DRAFT] Definition of API function `ara::diag::SovdServiceValidation::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdServiceValidation</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; Offer ();</code>	
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	<code>DiagOfferErrc::kAlready Offered</code>	This service was already offered.
	<code>DiagOfferErrc::k ConfigurationMismatch</code>	Implementation does not fit to the configuration.
<b>Description:</b>	Enable the DM to forward request messages to this handler.	

](RS\_Diag\_04256)

### 8.3.3.3.6 SovdServiceValidation::StopOffer function

[SWS\_DM\_01826][DRAFT] Definition of API function `ara::diag::SovdServiceValidation::StopOffer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdServiceValidation</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	<code>void StopOffer ();</code>	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Disable the forwarding of request messages from DM.	

](RS\_Diag\_04256)

### 8.3.3.3.7 SovdServiceValidation::Validate function

#### [SWS\_DM\_01827]{DRAFT} Definition of API function ara::diag::SovdServiceValidation::Validate

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_service_validation.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdServiceValidation</a>	
<b>Symbol:</b>	Validate(SovdRequestMethod requestMethod, const MetaInfo &metaInfo)	
<b>Syntax:</b>	virtual ara::core::Future< void > Validate (SovdRequestMethod requestMethod, const MetaInfo &metaInfo);	
<b>Parameters (in):</b>	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).
<b>Return value:</b>	ara::core::Future< void >	Value if the request is allowed, or an error.
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagSovdErrc::kSovdServerMisconfigured	If there is some fatal error in the setup of the application.
	DiagSovdErrc::kPreconditionNotFulfilled	If the precondition for the accessing the resource is not fulfilled.
	DiagSovdErrc::kTemporarilyNotAvailable	If the request currently can't be served.
	DiagSovdErrc::kInsufficientAccessRights	Or any other DiagSovdErrc for an SOVD-specific error message.
	DiagUdsNrcErrc::kRpmTooHigh	Or any other DiagUdsNrcErrc in case this is an error standardized in UDS.
<b>Description:</b>	Called for every SOVD request message. Will be called multiple times if the request is for a resource collection: <ul style="list-style-type: none"> <li>• once for the resource collection,</li> <li>• once for each resource in the resource collection.</li> </ul>	

|(RS\_Diag\_04256)

### 8.3.3.4 Http Redirect type

#### 8.3.3.4.1 HttpRedirect struct

#### [SWS\_DM\_01828]{DRAFT} Definition of API class ara::diag::HttpRedirect

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_http_redirect.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	HttpRedirect
<b>Syntax:</b>	struct HttpRedirect {...};





<b>Description:</b>	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.
---------------------	---

|(RS\_Diag\_04266, RS\_Diag\_04256)

### 8.3.3.4.2 HttpRedirect URL

[SWS\_DM\_01829]{DRAFT} Definition of API variable `ara::diag::HttpRedirect::url`

[

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_http_redirect.h"
<b>Scope:</b>	<a href="#">struct ara::diag::HttpRedirect</a>
<b>Symbol:</b>	url
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String url;
<b>Description:</b>	Specifies the url where the request shall be redirected.

|(RS\_Diag\_04266, RS\_Diag\_04256)

### 8.3.3.5 SOVD Bulk Data

#### 8.3.3.5.1 SovdBulkData

[SWS\_DM\_01841]{DRAFT} Definition of API class `ara::diag::SovdBulkData` [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	SovdBulkData
<b>Syntax:</b>	class SovdBulkData {...};
<b>Description:</b>	Generic SOVD Bulk Data interface.

|(RS\_Diag\_04266)

### 8.3.3.5.2 SovdBulkData::BulkDataMetaDataDescriptor type

#### [SWS\_DM\_01830]{DRAFT} Definition of API class ara::diag::SovdBulkData::BulkDataMetaDataDescriptor [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdBulkData</a>
<b>Symbol:</b>	BulkDataMetaDataDescriptor
<b>Syntax:</b>	struct BulkDataMetaDataDescriptor {...};
<b>Description:</b>	Definition of bulk data Meta Data Descriptor.

]([RS\\_Diag\\_04266](#))

#### [SWS\_DM\_01831]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::id [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</a>
<b>Symbol:</b>	id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String id;
<b>Description:</b>	Specifies the bulk data id.

]([RS\\_Diag\\_04266](#))

#### [SWS\_DM\_01832]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::mimetype [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</a>
<b>Symbol:</b>	mimetype
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String mimetype;
<b>Description:</b>	Specifies the bulk data mimetype.

]([RS\\_Diag\\_04266](#))

#### [SWS\_DM\_01833]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDataDescriptor::name [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDataDescriptor</a>
<b>Symbol:</b>	name





<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String name;
<b>Description:</b>	Specifies the bulk data name.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01834]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::translation\_id** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	translation_id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String translation_id;
<b>Description:</b>	Specifies the bulk data translation_id.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01835]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::size** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	size
<b>Type:</b>	ara::core::Optional< std::uint64_t >
<b>Syntax:</b>	ara::core::Optional<std::uint64_t> size;
<b>Description:</b>	Specifies the bulk data size in Byte.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01836]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::file\_name** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	file_name
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String file_name;
<b>Description:</b>	Specifies the filename.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01837]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::creation\_date** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	creation_date
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String creation_date;
<b>Description:</b>	Specifies the bulk data creation_date.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01838]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::last\_modified** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	last_modified
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String last_modified;
<b>Description:</b>	Specifies the bulk data last_modified.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01839]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::hash** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	hash
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String hash;
<b>Description:</b>	Specifies the bulk data hash.

]([RS\\_Diag\\_04266](#))

**[SWS\_DM\_01840]{DRAFT} Definition of API variable ara::diag::SovdBulkData::BulkDataMetaDescriptor::hash\_algorithm** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::BulkDataMetaDescriptor</a>
<b>Symbol:</b>	hash_algorithm
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String hash_algorithm;
<b>Description:</b>	Specifies the bulk data hash_algorithm.

|(RS\_Diag\_04266)

### 8.3.3.5.3 SovdBulkData::DeleteByCategoryResult type

[SWS\_DM\_01926]{DRAFT} **Definition of API class ara::diag::SovdBulkData::DeleteByCategoryResult** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdBulkData</a>
<b>Symbol:</b>	DeleteByCategoryResult
<b>Syntax:</b>	struct DeleteByCategoryResult { ...};
<b>Description:</b>	Definition of the result type for the "Delete All Bulk Data Defined by Category" method .

|(RS\_Diag\_04266)

[SWS\_DM\_01842]{DRAFT} **Definition of API variable ara::diag::SovdBulkData::DeleteByCategoryResult::deleted\_items** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::DeleteByCategoryResult</a>
<b>Symbol:</b>	deleted_items
<b>Type:</b>	ara::core::Vector< BulkDataMetaDataDescriptor >
<b>Syntax:</b>	ara::core::Vector<BulkDataMetaDataDescriptor> deleted_items;
<b>Description:</b>	List of bulk data ids that were successfully deleted .

|(RS\_Diag\_04266)

[SWS\_DM\_01843]{DRAFT} **Definition of API variable ara::diag::SovdBulkData::DeleteByCategoryResult::failed\_items** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::DeleteByCategoryResult</a>
<b>Symbol:</b>	failed_items
<b>Type:</b>	ara::core::Vector< DeletionError >
<b>Syntax:</b>	ara::core::Vector<DeletionError> failed_items;
<b>Description:</b>	A list of DeletionErrors that contains the bulk data instances with corresponding deletion errors .

|(RS\_Diag\_04266)

### 8.3.3.5.4 SovdBulkData::DeletionError type

[SWS\_DM\_01844]{DRAFT} **Definition of API class ara::diag::SovdBulkData::DeletionError** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdBulkData</a>
<b>Symbol:</b>	DeletionError
<b>Syntax:</b>	struct DeletionError {...};
<b>Description:</b>	Definition of DeletionError for a single bulk-data instance .

]([RS\\_Diag\\_04266](#))

[SWS\_DM\_01845]{DRAFT} **Definition of API variable ara::diag::SovdBulkData::DeletionError::item** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::DeletionError</a>
<b>Symbol:</b>	item
<b>Type:</b>	<a href="#">BulkDataMetaDataDescriptor</a>
<b>Syntax:</b>	BulkDataMetaDataDescriptor item;
<b>Description:</b>	bulk data that could not be deleted

]([RS\\_Diag\\_04266](#))

[SWS\_DM\_01846]{DRAFT} **Definition of API variable ara::diag::SovdBulkData::DeletionError::error** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdBulkData::DeletionError</a>
<b>Symbol:</b>	error
<b>Type:</b>	<a href="#">ara::core::ErrorCode</a>
<b>Syntax:</b>	ara::core::ErrorCode error;
<b>Description:</b>	Corresponding error, that specifies why the bulk data could not be deleted .

]([RS\\_Diag\\_04266](#))



### 8.3.3.5.5 SovdBulkData::SovdBulkData constructor

#### [SWS\_DM\_01847]{DRAFT} Definition of API function ara::diag::SovdBulkData::SovdBulkData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	SovdBulkData(ara::core::InstanceSpecifier const &instanceSpecifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	SovdBulkData (ara::core::InstanceSpecifier const &instanceSpecifier, ReentrancyType reentrancyType) noexcept;	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdBulkData interface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructor for the SovdBulkData Interface.	

](RS\_Diag\_04266)

#### [SWS\_DM\_01848]{DRAFT} Definition of API function ara::diag::SovdBulkData::SovdBulkData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	SovdBulkData(SovdBulkData const &)	
<b>Syntax:</b>	SovdBulkData (SovdBulkData const &)=delete;	
<b>Description:</b>	SovdBulkData shall be a single not copy-able instance.	
<b>Visibility:</b>	protected	

](RS\_Diag\_04266)

#### [SWS\_DM\_01849]{DRAFT} Definition of API function ara::diag::SovdBulkData::SovdBulkData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	SovdBulkData(SovdBulkData &&other)	
<b>Syntax:</b>	SovdBulkData (SovdBulkData &&other) noexcept;	
<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Move constructs an instance of SovdBulkData.	
<b>Visibility:</b>	protected	

](RS\_Diag\_04266)

### 8.3.3.5.6 SovdBulkData::~SovdBulkData destructor

[SWS\_DM\_01852]{DRAFT} Definition of API function `ara::diag::SovdBulkData::~~SovdBulkData` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<code>class ara::diag::SovdBulkData</code>
<b>Symbol:</b>	<code>~SovdBulkData()</code>
<b>Syntax:</b>	<code>virtual ~SovdBulkData () noexcept=default;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SovdBulkData.

]([RS\\_Diag\\_04262](#))

### 8.3.3.5.7 SovdBulkData::copy assignment operator

[SWS\_DM\_01850]{DRAFT} Definition of API function `ara::diag::SovdBulkData::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"
<b>Scope:</b>	<code>class ara::diag::SovdBulkData</code>
<b>Symbol:</b>	<code>operator=(SovdBulkData const &amp;)</code>
<b>Syntax:</b>	<code>SovdBulkData &amp; operator= (SovdBulkData const &amp;)=delete;</code>
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	SovdBulkData shall be a single not assignable instance.
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04266](#))

### 8.3.3.5.8 SovdBulkData::move assignment operator

[SWS\_DM\_01851]{DRAFT} Definition of API function `ara::diag::SovdBulkData::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	<code>class ara::diag::SovdBulkData</code>	
<b>Symbol:</b>	<code>operator=(SovdBulkData &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>SovdBulkData &amp; operator= (SovdBulkData &amp;&amp;other) &amp; noexcept;</code>	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	SovdBulkData &	Reference to self
<b>Exception Safety:</b>	noexcept	





<b>Thread Safety:</b>	unsafe
<b>Description:</b>	Move assigns an instance of SovdBulkData.
<b>Visibility:</b>	protected

|(RS\_Diag\_04266)

### 8.3.3.5.9 SovdBulkData::GetBulkDataMetaData function

[SWS\_DM\_01853]{DRAFT} Definition of API function ara::diag::SovdBulkData::GetBulkDataMetaData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	GetBulkDataMetaData(ara::core::String bulkDataCategory, MetaInfo const &metaInfo, ara::diag::ReleaseHandler releaseHandler, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; ara::core::Span&lt; BulkDataMetaDescriptor &gt; &gt; GetBulkDataMetaData (ara::core::String bulkDataCategory, MetaInfo const &amp;metaInfo, ara::diag::ReleaseHandler releaseHandler, CancellationHandler cancellationHandler) noexcept=0;</pre>	
<b>Parameters (in):</b>	bulkDataCategory	Bulk data category to be read
	metaInfo	Contains additional meta information
	releaseHandler	releaseHandler to notify when the result Span can be modified again
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< ara::core::Span< BulkDataMetaDescriptor > >	A future with a span of all bulk data instances of the requested bulk data category. The span shall not be modified until the release Handler is called.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Read Bulk Data Meta Data".	

|(RS\_Diag\_04266)

### 8.3.3.5.10 SovdBulkData::RequestBulkDataDownload function

[SWS\_DM\_01873]{DRAFT} Definition of API function ara::diag::SovdBulkData::RequestBulkDataDownload [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	





<b>Symbol:</b>	RequestBulkDataDownload(ara::core::String bulkDataCategory, ara::core::String bulkDataId, ara::core::Vector< ara::core::String > acceptHeader, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< ara::core::Variant< std::tuple< DataTransferReadSession, BulkDataMetaDataDescriptor >, HttpRedirect > > RequestBulkDataDownload (ara::core::String bulkDataCategory, ara::core::String bulkDataId, ara::core::Vector< ara::core::String > acceptHeader, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	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
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>DIRECTION NOT DEFINED</b>	acceptHeader	–
<b>Return value:</b>	ara::core::Future< ara::core::Variant< std::tuple< DataTransferReadSession, BulkDataMetaDataDescriptor >, HttpRedirect > >	A future with either an instance of a "SOVD file reading session" with the corresponding BulkDataMetaDataDescriptor, an HTTP redirect or a SOVD Error @limitations no signature, no compression etc.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Download Bulk Data".	

|(RS\_Diag\_04266)

### 8.3.3.5.11 SovdBulkData::RequestBulkDataUpload function

[SWS\_DM\_01854]{DRAFT} **Definition of API function ara::diag::SovdBulkData::RequestBulkDataUpload** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	RequestBulkDataUpload(ara::core::String bulkDataCategory, BulkDataMetaDataDescriptor bulkDataMetaData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< std::tuple< DataTransferWriteSession, BulkDataMetaDataDescriptor > > RequestBulkDataUpload (ara::core::String bulkDataCategory, BulkDataMetaDataDescriptor bulkDataMetaData, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	bulkDataCategory	Bulk data category of which a new bulk data instance shall be uploaded
	bulkDataMetaData	BulkDataMetaDataDescriptor where the available information from the SOVD POST Request headers are prefilled
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled





<b>Return value:</b>	ara::core::Future< std::tuple< DataTransfer WriteSession, BulkData MetaDataDescriptor > >	A future with either an instance of a "SOVD file writing session" with the corresponding BulkDataMetaDataDescriptor or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Upload Bulk Data".	

](RS\_Diag\_04266)

### 8.3.3.5.12 SovdBulkData::DeleteAllBulkData function

[SWS\_DM\_01855]{DRAFT} Definition of API function ara::diag::SovdBulkData::DeleteAllBulkData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	DeleteAllBulkData(ara::core::String bulkDataCategory, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< DeleteByCategoryResult > DeleteAllBulkData (ara::core::String bulkDataCategory, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	bulkDataCategory	Bulk data category of which all instances shall be deleted
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< DeleteByCategoryResult >	A future with a DeleteByCategoryResult or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Delete All Bulk Data Defined by Category".	

](RS\_Diag\_04266)

### 8.3.3.5.13 SovdBulkData::DeleteSpecificBulkData function

[SWS\_DM\_01856]{DRAFT} Definition of API function ara::diag::SovdBulkData::DeleteSpecificBulkData [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	DeleteSpecificBulkData(ara::core::String bulkDataCategory, ara::core::String bulkDataId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	





<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; void &gt; DeleteSpecificBulkData (ara::core::String bulkDataCategory, ara::core::String bulkDataId, MetaInfo const &amp;metaInfo, CancellationHandler cancellationHandler) noexcept=0;</pre>	
<b>Parameters (in):</b>	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	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< void >	A future with either a void data result for a successful deletion or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Delete Specific Bulk Data Resource".	

](RS\_Diag\_04266)

#### 8.3.3.5.14 SovdBulkData::Offer function

[SWS\_DM\_01857]{DRAFT} Definition of API function ara::diag::SovdBulkData::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer () noexcept;	
<b>Return value:</b>	ara::core::Result< void >	Positive result if service was offered, error if offer failed
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Errors:</b>	DiagOfferErrc::kConfigurationMismatch	if handle is an invalid handle
	DiagOfferErrc::kAlreadyOffered	if service has been offered already
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

](RS\_Diag\_04266)

#### 8.3.3.5.15 SovdBulkData::StopOffer function

[SWS\_DM\_01858]{DRAFT} Definition of API function ara::diag::SovdBulkData::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_bulk_data.h"	
<b>Scope:</b>	class ara::diag::SovdBulkData	





<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	<code>void StopOffer () const noexcept;</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	noexcept
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04266](#))

### 8.3.3.6 SOVD Software Update

#### 8.3.3.6.1 SovdSwUpdate

[SWS\_DM\_01874][DRAFT] Definition of API class `ara::diag::SovdSwUpdate` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/sovd_sw_update.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>SovdSwUpdate</code>
<b>Syntax:</b>	<code>class SovdSwUpdate {...};</code>
<b>Description:</b>	Class to implement the SovdSwUpdate interface.

]([RS\\_Diag\\_04265](#))

#### 8.3.3.6.2 `diag::SovdSwUpdate::SovdUpdatePhase` enum

[SWS\_DM\_01875][DRAFT] Definition of API enum `ara::diag::SovdUpdatePhase` [

<b>Kind:</b>	enumeration				
<b>Header file:</b>	<code>#include "ara/diag/sovd_sw_update.h"</code>				
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>				
<b>Scope:</b>	namespace <code>ara::diag</code>				
<b>Symbol:</b>	<code>SovdUpdatePhase</code>				
<b>Underlying type:</b>	<code>std::uint8_t</code>				
<b>Syntax:</b>	<code>enum class SovdUpdatePhase : std::uint8_t {...};</code>				
<b>Values:</b>	<table border="1"> <tr> <td><code>kPrepare</code></td> <td>–</td> </tr> <tr> <td><code>kExecute</code></td> <td>&lt; The Update is in preparation.&lt; The Update is in execution</td> </tr> </table>	<code>kPrepare</code>	–	<code>kExecute</code>	< The Update is in preparation.< The Update is in execution
<code>kPrepare</code>	–				
<code>kExecute</code>	< The Update is in preparation.< The Update is in execution				
<b>Description:</b>	Phase type for a SOVD update.				

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.3 diag::SovdSwUpdate::SovdUpdateStatus enum

[SWS\_DM\_01876]{DRAFT} Definition of API enum ara::diag::SovdUpdateStatus [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag	
<b>Symbol:</b>	SovdUpdateStatus	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class SovdUpdateStatus : std::uint8_t {...};	
<b>Values:</b>	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.
<b>Description:</b>	Status type for a SOVD update .	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.4 SovdSwUpdate::UpdatePackageDetails type

[SWS\_DM\_01877]{DRAFT} Definition of API class ara::diag::SovdSwUpdate::UpdatePackageDetails [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>
<b>Symbol:</b>	UpdatePackageDetails
<b>Syntax:</b>	struct UpdatePackageDetails {...};
<b>Description:</b>	Definition of Update Package Details .

]([RS\\_Diag\\_04265](#))

[SWS\_DM\_01878]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::id [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String id;
<b>Description:</b>	Specifies the update package id.

]([RS\\_Diag\\_04265](#))



**[SWS\_DM\_01879]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageDetails::name` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	name
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String name;
<b>Description:</b>	Specifies the update package update_name.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01881]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageDetails::translation_id` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	translation_id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String translation_id;
<b>Description:</b>	Specifies the update package update_translation_id.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01882]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageDetails::automated` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	automated
<b>Type:</b>	ara::core::Optional< bool >
<b>Syntax:</b>	ara::core::Optional<bool> automated;
<b>Description:</b>	Specifies the update package attribute "automated".

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01883]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageDetails::origin` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	origin
<b>Type:</b>	ara::core::Vector< ara::core::String >
<b>Syntax:</b>	ara::core::Vector<ara::core::String> origin;
<b>Description:</b>	Specifies the update package origin.

](RS\_Diag\_04265)

**[SWS\_DM\_01884]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::notes** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	notes
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String notes;
<b>Description:</b>	Specifies the update package notes.

](RS\_Diag\_04265)

**[SWS\_DM\_01885]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::notes\_translation\_id** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	notes_translation_id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String notes_translation_id;
<b>Description:</b>	Specifies the update package notes_translation_id.

](RS\_Diag\_04265)

**[SWS\_DM\_01886]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::user\_activity** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	user_activity
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String user_activity;
<b>Description:</b>	Specifies the update package user_activity.

](RS\_Diag\_04265)

**[SWS\_DM\_01887]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::user\_activity\_translation\_id** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	user_activity_translation_id





<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String user_activity_translation_id;
<b>Description:</b>	Specifies the update package user_activity_translation_id.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01888]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::preconditions** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	preconditions
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String preconditions;
<b>Description:</b>	Specifies the update package preconditions.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01889]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::preconditions\_translation\_id** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	preconditions_translation_id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String preconditions_translation_id;
<b>Description:</b>	Specifies the update package preconditions_translation_id.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01890]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::execution\_conditions** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	execution_conditions
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String execution_conditions;
<b>Description:</b>	Specifies the update package execution_conditions.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01891]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::duration** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	duration
<b>Type:</b>	ara::core::Optional< std::uint64_t >
<b>Syntax:</b>	ara::core::Optional<std::uint64_t> duration;
<b>Description:</b>	Specifies the update package duration in seconds.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01893]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::size** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	size
<b>Type:</b>	std::uint64_t
<b>Syntax:</b>	std::uint64_t size;
<b>Description:</b>	Specifies the update package size in Bytes .

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01894]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::updated\_components** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	updated_components
<b>Type:</b>	ara::core::Vector< ara::core::String >
<b>Syntax:</b>	ara::core::Vector<ara::core::String> updated_components;
<b>Description:</b>	Specifies the update package updated_components.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01895]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageDetails::affected\_components** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageDetails</a>
<b>Symbol:</b>	affected_components
<b>Type:</b>	ara::core::Vector< ara::core::String >
<b>Syntax:</b>	ara::core::Vector<ara::core::String> affected_components;
<b>Description:</b>	Specifies the update package affected_components.

](RS\_Diag\_04265)

### 8.3.3.6.5 SovdSwUpdate::SubProgress type

[SWS\_DM\_01896]{DRAFT} Definition of API class ara::diag::SovdSwUpdate::SubProgress [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>
<b>Symbol:</b>	SubProgress
<b>Syntax:</b>	struct SubProgress {...};
<b>Description:</b>	Definition of Sovd Progress type.

](RS\_Diag\_04265)

[SWS\_DM\_01898]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::SubProgress::entity [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::SubProgress</a>
<b>Symbol:</b>	entity
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String entity;
<b>Description:</b>	Specifies the entity that is currently being updated.

](RS\_Diag\_04265)

[SWS\_DM\_01899]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::SubProgress::status [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::SubProgress</a>
<b>Symbol:</b>	status
<b>Type:</b>	SovdUpdateStatus
<b>Syntax:</b>	SovdUpdateStatus status;
<b>Description:</b>	Specifies the Update Status.

](RS\_Diag\_04265)

**[SWS\_DM\_01900]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::SubProgress::progress` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct <code>ara::diag::SovdSwUpdate::SubProgress</code></a>
<b>Symbol:</b>	progress
<b>Type:</b>	<code>ara::core::Optional&lt; std::uint8_t &gt;</code>
<b>Syntax:</b>	<code>ara::core::Optional&lt;std::uint8_t&gt; progress;</code>
<b>Description:</b>	Specifies the Update progress in percent.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01901]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::SubProgress::error` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct <code>ara::diag::SovdSwUpdate::SubProgress</code></a>
<b>Symbol:</b>	error
<b>Type:</b>	<code>ara::core::ErrorCode</code>
<b>Syntax:</b>	<code>ara::core::ErrorCode error;</code>
<b>Description:</b>	Specifies the error if status is failed.

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.6 SovdSwUpdate::UpdatePackageStatus type

**[SWS\_DM\_01902]{DRAFT} Definition of API class `ara::diag::SovdSwUpdate::UpdatePackageStatus` [**

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdSwUpdate</code></a>
<b>Symbol:</b>	UpdatePackageStatus
<b>Syntax:</b>	<code>struct UpdatePackageStatus { ... };</code>
<b>Description:</b>	Definition of Update Status.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01903]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::phase** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	phase
<b>Type:</b>	SovdUpdatePhase
<b>Syntax:</b>	SovdUpdatePhase phase;
<b>Description:</b>	Specifies the Update Status phase.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01904]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::status** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	status
<b>Type:</b>	SovdUpdateStatus
<b>Syntax:</b>	SovdUpdateStatus status;
<b>Description:</b>	Specifies the status of the Update phase.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01905]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::progress** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	progress
<b>Type:</b>	ara::core::Optional< std::uint8_t >
<b>Syntax:</b>	ara::core::Optional<std::uint8_t> progress;
<b>Description:</b>	Specifies the progress of the Update phase in percentage.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01906]{DRAFT} Definition of API variable ara::diag::SovdSwUpdate::UpdatePackageStatus::subprogress** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	subprogress
<b>Type:</b>	ara::core::Vector< SubProgress >
<b>Syntax:</b>	ara::core::Vector<SubProgress> subprogress;
<b>Description:</b>	Specifies the progress of the Update phase in percent.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01907]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageStatus::step` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	step
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String step;
<b>Description:</b>	Specifies the current step of the Update.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01908]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageStatus::step_translation_id` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	step_translation_id
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String step_translation_id;
<b>Description:</b>	Specifies the step_translation_id.

]([RS\\_Diag\\_04265](#))

**[SWS\_DM\_01909]{DRAFT} Definition of API variable `ara::diag::SovdSwUpdate::UpdatePackageStatus::error` [**

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdSwUpdate::UpdatePackageStatus</a>
<b>Symbol:</b>	error
<b>Type:</b>	ara::core::Optional< ara::diag::DiagSovdErrc >
<b>Syntax:</b>	ara::core::Optional<ara::diag::DiagSovdErrc> error;
<b>Description:</b>	Specifies the error if status is failed.

]([RS\\_Diag\\_04265](#))



### 8.3.3.6.7 SovdSwUpdate::SovdSwUpdate constructor

[SWS\_DM\_01910]{DRAFT} **Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	SovdSwUpdate(ara::core::InstanceSpecifier const &instanceSpecifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	SovdSwUpdate (ara::core::InstanceSpecifier const &instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdSwUpdate service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructor for SovdSwUpdate.	

]([RS\\_Diag\\_04265](#))

[SWS\_DM\_01912]{DRAFT} **Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	SovdSwUpdate(SovdSwUpdate const &)	
<b>Syntax:</b>	SovdSwUpdate (SovdSwUpdate const &)=delete;	
<b>Description:</b>	SovdSwUpdate shall be a single not copy-able instance.	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04265](#))

[SWS\_DM\_01913]{DRAFT} **Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	SovdSwUpdate(SovdSwUpdate &&other)	
<b>Syntax:</b>	SovdSwUpdate (SovdSwUpdate &&other) noexcept;	
<b>Parameters (out):</b>	other	The other object
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Move constructs an instance of SovdSwUpdate.	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.8 SovdSwUpdate::~SovdSwUpdate destructor

[SWS\_DM\_01911]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::~~SovdSwUpdate` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>
<b>Symbol:</b>	<code>~SovdSwUpdate()</code>
<b>Syntax:</b>	<code>virtual ~SovdSwUpdate () noexcept=default;</code>
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of class SovdSwUpdate.

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.9 SovdSwUpdate::copy assignment operator

[SWS\_DM\_01914]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>
<b>Symbol:</b>	<code>operator=(SovdSwUpdate const &amp;)</code>
<b>Syntax:</b>	<code>SovdSwUpdate &amp; operator= (SovdSwUpdate const &amp;)=delete;</code>
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	SovdSwUpdate shall be a single not assignable instance.
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.10 SovdSwUpdate::move assignment operator

[SWS\_DM\_01915]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	<code>operator=(SovdSwUpdate &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>SovdSwUpdate &amp; operator= (SovdSwUpdate &amp;&amp;other) &amp; noexcept;</code>	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	SovdSwUpdate &	Reference to self
<b>Exception Safety:</b>	noexcept	





<b>Thread Safety:</b>	unsafe
<b>Description:</b>	Move assigns an instance of SovdSwUpdate.
<b>Visibility:</b>	protected

](RS\_Diag\_04265)

### 8.3.3.6.11 SovdSwUpdate::GetAllUpdates function

[SWS\_DM\_01916]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::GetAllUpdates` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	GetAllUpdates(ara::core::String targetVersion, ara::core::String origin, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; ara::core::Vector&lt; ara::core::String &gt; &gt; GetAllUpdates (ara::core::String targetVersion, ara::core::String origin, MetaInfo const &amp;metaInfo, CancellationHandler cancellation Handler) noexcept=0;</pre>	
<b>Parameters (in):</b>	targetVersion	Optional parameter to be used if the target-version query parameter is used in the request. If the parameter is not set the String shall be left empty. set in the request the String shall be left empty
	origin	Optional parameter to be used if the origin query parameter is used in the request. If the parameter is not set the String shall be left empty
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	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.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Retrieve List of All Updates".	

](RS\_Diag\_04265)

### 8.3.3.6.12 SovdSwUpdate::GetUpdatePackageDetails function

[SWS\_DM\_01917]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::GetUpdatePackageDetails` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	





<b>Symbol:</b>	GetUpdatePackageDetails(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< UpdatePackageDetails > GetUpdatePackageDetails (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id of the update to be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< UpdatePackageDetails >	A future with the details of the requested update package or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Get Details of Update".	

|(RS\_Diag\_04265)

### 8.3.3.6.13 SovdSwUpdate::PutUpdatePackageAutomated function

[SWS\_DM\_01918]{DRAFT} **Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackageAutomated** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	class ara::diag::SovdSwUpdate	
<b>Symbol:</b>	PutUpdatePackageAutomated(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > PutUpdatePackageAutomated (ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id of the update to be put to automated
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< void >	A future with either a positive result or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Automated Installation of an Update".	

|(RS\_Diag\_04265)

### 8.3.3.6.14 SovdSwUpdate::PrepareUpdatePackage function

[SWS\_DM\_01919]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::PrepareUpdatePackage` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	PrepareUpdatePackage(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > PrepareUpdatePackage (ara::core::String updatePackageId, <a href="#">MetaInfo</a> const &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id of the update to be prepared
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< void >	A future with either a positive result or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Prepare Installation of an Update".	

] ([RS\\_Diag\\_04265](#))

### 8.3.3.6.15 SovdSwUpdate::ExecuteUpdatePackage function

[SWS\_DM\_01920]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::ExecuteUpdatePackage` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	ExecuteUpdatePackage(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > ExecuteUpdatePackage (ara::core::String updatePackageId, <a href="#">MetaInfo</a> const &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id of the update to be executed
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< void >	A future with either a positive results or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Execute Installation of an Update".	

] ([RS\\_Diag\\_04265](#))

### 8.3.3.6.16 SovdSwUpdate::GetUpdatePackageStatus function

[SWS\_DM\_01921]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::GetUpdatePackageStatus` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	GetUpdatePackageStatus(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">UpdatePackageStatus</a> > GetUpdatePackageStatus (ara::core::String updatePackageId, <a href="#">MetaInfo</a> const &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id of which the status shall be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< UpdatePackageStatus >	A future with either the current UpdatePackageStatus or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Get Status of an Update".	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.17 SovdSwUpdate::DeleteUpdatePackage function

[SWS\_DM\_01922]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::DeleteUpdatePackage` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	DeleteUpdatePackage(ara::core::String updatePackageId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< void > DeleteUpdatePackage (ara::core::String updatePackageId, <a href="#">MetaInfo</a> const &metaInfo, <a href="#">CancellationHandler</a> cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	updatePackageId	update-package-id to be deleted
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< void >	A future with either a positive results or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Delete Update Package from an SOVD Server".	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.18 SovdSwUpdate::RequestUpdatePackageRegistration function

[SWS\_DM\_01923]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::RequestUpdatePackageRegistration` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	RequestUpdatePackageRegistration(MetaInfo const &metaInfo, ara::core::Optional< uint64_t > expectedSize, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	<pre>virtual ara::core::Future&lt; std::tuple&lt; DataTransferWriteSession, ara::core::String &gt; &gt; RequestUpdatePackageRegistration (MetaInfo const &amp;metaInfo, ara::core::Optional&lt; uint64_t &gt; expectedSize, Cancellation Handler cancellationHandler) noexcept=0;</pre>	
<b>Parameters (in):</b>	metaInfo	Contains additional meta information
	expectedSize	Specifies the expected size if available in the request
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< std::tuple< DataTransfer WriteSession, ara::core::String > >	A future with either an instance of a "SOVD file writing session" with the corresponding update package id or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Register an Update at the SOVD Server".	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.19 SovdSwUpdate::Offer function

[SWS\_DM\_01924]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_sw_update.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdSwUpdate</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	<pre>ara::core::Result&lt; void &gt; Offer () noexcept;</pre>	
<b>Return value:</b>	ara::core::Result< void >	Positive result if service was offered, error if offer failed
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Errors:</b>	DiagOfferErrc::k ConfigurationMismatch	if handle is an invalid handle
	DiagOfferErrc::kAlready Offered	if service has been offered already
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_Diag\\_04265](#))

### 8.3.3.6.20 SovdSwUpdate::StopOffer function

[SWS\_DM\_01925]{DRAFT} Definition of API function `ara::diag::SovdSwUpdate::StopOffer` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/sovd_sw_update.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdSwUpdate</code></a>
<b>Symbol:</b>	<code>StopOffer()</code>
<b>Syntax:</b>	<code>void StopOffer () const noexcept;</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	<code>noexcept</code>
<b>Thread Safety:</b>	<code>unsafe</code>
<b>Description:</b>	This <code>StopOffer</code> will disable the forwarding of request messages from DM.

]([RS\\_Diag\\_04265](#))

### 8.3.3.7 SOVD Configuration

#### 8.3.3.7.1 SovdConfiguration

[SWS\_DM\_01860]{DRAFT} Definition of API class `ara::diag::SovdConfiguration` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/sovd_configuration.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>SovdConfiguration</code>
<b>Syntax:</b>	<code>class SovdConfiguration {...};</code>
<b>Description:</b>	Generic SOVD Configuration interface.

]([RS\\_Diag\\_04261](#))

#### 8.3.3.7.2 SovdConfiguration::SovdConfigurationMetaInfo type

[SWS\_DM\_01861]{DRAFT} Definition of API class `ara::diag::SovdConfiguration::SovdConfigurationMetaInfo` [

<b>Kind:</b>	struct
<b>Header file:</b>	<code>#include "ara/diag/sovd_configuration.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	<a href="#">class <code>ara::diag::SovdConfiguration</code></a>
<b>Symbol:</b>	<code>SovdConfigurationMetaInfo</code>
<b>Syntax:</b>	<code>struct SovdConfigurationMetaInfo {...};</code>
<b>Description:</b>	MetaInfo for Transfer of SOVD Configurations.



|(RS\_Diag\_04261)

**[SWS\_DM\_01862]{DRAFT} Definition of API variable ara::diag::SovdConfiguration::SovdConfigurationMetalInfo::size** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdConfiguration::SovdConfigurationMetalInfo</a>
<b>Symbol:</b>	size
<b>Type:</b>	ara::core::Optional< std::uint64_t >
<b>Syntax:</b>	ara::core::Optional<std::uint64_t> size;
<b>Description:</b>	Specifies the size of the configuration data .

|(RS\_Diag\_04261)

**[SWS\_DM\_01863]{DRAFT} Definition of API variable ara::diag::SovdConfiguration::SovdConfigurationMetalInfo::mimetype** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">struct ara::diag::SovdConfiguration::SovdConfigurationMetalInfo</a>
<b>Symbol:</b>	mimetype
<b>Type:</b>	ara::core::String
<b>Syntax:</b>	ara::core::String mimetype;
<b>Description:</b>	Specifies the configuration mimetype .

|(RS\_Diag\_04261)

### 8.3.3.7.3 SovdConfiguration::SovdConfiguration constructor

**[SWS\_DM\_01864]{DRAFT} Definition of API function ara::diag::SovdConfiguration::SovdConfiguration** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>	
<b>Symbol:</b>	SovdConfiguration(ara::core::InstanceSpecifier const &instanceSpecifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	SovdConfiguration (ara::core::InstanceSpecifier const &instanceSpecifier, <a href="#">ReentrancyType</a> reentrancyType) noexcept;	
<b>Parameters (in):</b>	instanceSpecifier	InstanceSpecifier to a PortPrototype of a SovdConfiguration interface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Constructor for the SovdConfiguration Interface.	

|(RS\_Diag\_04261)

**[SWS\_DM\_01866]{DRAFT} Definition of API function ara::diag::SovdConfiguration::SovdConfiguration** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>
<b>Symbol:</b>	SovdConfiguration(SovdConfiguration const &)
<b>Syntax:</b>	SovdConfiguration (SovdConfiguration const &)=delete;
<b>Description:</b>	SovdConfiguration shall be a single not copy-able instance. .
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04261](#))

**[SWS\_DM\_01867]{DRAFT} Definition of API function ara::diag::SovdConfiguration::SovdConfiguration** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>
<b>Symbol:</b>	SovdConfiguration(SovdConfiguration &&other)
<b>Syntax:</b>	SovdConfiguration (SovdConfiguration &&other) noexcept;
<b>Parameters (out):</b>	other   The other object
<b>Exception Safety:</b>	noexcept
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	Move constructs an instance of SovdConfiguration.
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04261](#))

#### 8.3.3.7.4 SovdConfiguration::~SovdConfiguration destructor

**[SWS\_DM\_01865]{DRAFT} Definition of API function ara::diag::SovdConfiguration::~~SovdConfiguration** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>
<b>Symbol:</b>	~SovdConfiguration()
<b>Syntax:</b>	virtual ~SovdConfiguration () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of SovdConfiguration.

]([RS\\_Diag\\_04261](#))

### 8.3.3.7.5 SovdConfiguration::copy assignment operator

[SWS\_DM\_01868]{DRAFT} Definition of API function `ara::diag::SovdConfiguration::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>
<b>Symbol:</b>	<code>operator=(SovdConfiguration const &amp;)</code>
<b>Syntax:</b>	<code>SovdConfiguration &amp; operator= (SovdConfiguration const &amp;)=delete;</code>
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	SovdConfiguration shall be a single not assignable instance.
<b>Visibility:</b>	protected

]([RS\\_Diag\\_04261](#))

### 8.3.3.7.6 SovdConfiguration::move assignment operator

[SWS\_DM\_01869]{DRAFT} Definition of API function `ara::diag::SovdConfiguration::operator=` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"	
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>	
<b>Symbol:</b>	<code>operator=(SovdConfiguration &amp;&amp;other)</code>	
<b>Syntax:</b>	<code>SovdConfiguration &amp; operator= (SovdConfiguration &amp;&amp;other) &amp; noexcept;</code>	
<b>Parameters (out):</b>	other	The other object
<b>Return value:</b>	SovdConfiguration &	Reference to self
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Move assigns an instance of SovdConfiguration.	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04261](#))

### 8.3.3.7.7 SovdConfiguration::RequestGetConfiguration function

[SWS\_DM\_01870]{DRAFT} Definition of API function `ara::diag::SovdConfiguration::RequestGetConfiguration` [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"
<b>Scope:</b>	<a href="#">class ara::diag::SovdConfiguration</a>





<b>Symbol:</b>	RequestGetConfiguration(ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< std::tuple< DataTransferReadSession, Sovd ConfigurationMetaInfo > > RequestGetConfiguration (ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	configurationId	configuration-id of the configuration instance to be read
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	ara::core::Future< std::tuple< DataTransfer ReadSession, Sovd ConfigurationMetaInfo > >	A future with either an instance of a "SOVD file reading session" with corresponding SovdConfigurationMetaInfo or a SOVD Error
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Read Configuration".	

|(RS\_Diag\_04261)

### 8.3.3.7.8 SovdConfiguration::RequestPutConfiguration function

[SWS\_DM\_01871]{DRAFT} Definition of API function ara::diag::SovdConfiguration::RequestPutConfiguration [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"	
<b>Scope:</b>	class ara::diag::SovdConfiguration	
<b>Symbol:</b>	RequestPutConfiguration(ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler)	
<b>Syntax:</b>	virtual ara::core::Future< std::tuple< DataTransferWriteSession, Sovd ConfigurationMetaInfo > > RequestPutConfiguration (ara::core::String configurationId, MetaInfo const &metaInfo, CancellationHandler cancellationHandler) noexcept=0;	
<b>Parameters (in):</b>	configurationId	configuration-id of the configuration instance to be written
	metaInfo	Contains additional meta information
	cancellationHandler	Informs if the current conversation is canceled
<b>Return value:</b>	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
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Called for SOVD method "Write Configuration".	

|(RS\_Diag\_04261)

### 8.3.3.7.9 SovdConfiguration::Offer function

[SWS\_DM\_01872]{DRAFT} Definition of API function `ara::diag::SovdConfiguration::Offer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"	
<b>Scope:</b>	class <a href="#">ara::diag::SovdConfiguration</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	<code>ara::core::Result&lt; void &gt; Offer () noexcept;</code>	
<b>Return value:</b>	<code>ara::core::Result&lt; void &gt;</code>	Positive result if service was offered, error if offer failed
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Errors:</b>	<code>DiagOfferErrc::kConfigurationMismatch</code>	if handle is an invalid handle
	<code>DiagOfferErrc::kAlreadyOffered</code>	if service has been offered already
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_Diag\\_04261](#))

### 8.3.3.7.10 SovdConfiguration::StopOffer function

[SWS\_DM\_01930]{DRAFT} Definition of API function `ara::diag::SovdConfiguration::StopOffer` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/sovd_configuration.h"	
<b>Scope:</b>	class <a href="#">ara::diag::SovdConfiguration</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	<code>void StopOffer () const noexcept;</code>	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([RS\\_Diag\\_04261](#))

## A Mentioned Manifest Elements

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

<b>Class</b>	<b>ApApplicationError</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
<b>Note</b>	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform <b>Tags:</b> atp.recommendedPackage=ApplicationErrors			
<b>Base</b>	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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**

<b>Class</b>	<b>BaseType</b> (abstract)			
<b>Package</b>	M2::MSR::AsamHdo::BaseTypes			
<b>Note</b>	This abstract meta-class represents the ability to specify a platform dependent base type.			
<b>Base</b>	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Subclasses</b>	SwBaseType			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

**Table A.2: BaseType**

<b>Class</b>	<b>BaseTypeDirectDefinition</b>			
<b>Package</b>	M2::MSR::AsamHdo::BaseTypes			
<b>Note</b>	This BaseType is defined directly (as opposite to a derived BaseType)			
<b>Base</b>	ARObject, BaseTypeDefinition			
<b>Aggregated by</b>	BaseType.baseTypeDefinition			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class	BaseTypeDirectDefinition			
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. <b>Tags:</b> xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. <b>Tags:</b> xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. <b>Tags:</b> xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". <b>Tags:</b> xml.sequenceOffset=100
nativeDeclaration	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. <b>Tags:</b> xml.sequenceOffset=120

**Table A.3: BaseTypeDirectDefinition**

Class	CompuConst			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	ARObject			
Aggregated by	Compu.compuDefaultValue, <a href="#">CompuScale.compuInverseValue</a> , <a href="#">CompuScaleConstantContents.compuConst</a>			
Attribute	Type	Mult.	Kind	Note
compuConstContent	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

**Table A.4: CompuConst**

<b>Class</b>	<b>CompuConstFormulaContent</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>CompuConstContent</i>			
<b>Aggregated by</b>	<a href="#">CompuConst.compuConstContentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

**Table A.5: CompuConstFormulaContent**

<b>Class</b>	<b>CompuConstTextContent</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the textual content of a scale.			
<b>Base</b>	<i>ARObject</i> , <i>CompuConstContent</i>			
<b>Aggregated by</b>	<a href="#">CompuConst.compuConstContentType</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

**Table A.6: CompuConstTextContent**

<b>Class</b>	<b>CompuScale</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to specify one segment of a segmented computation method.			
<b>Base</b>	<i>ARObject</i>			
<b>Aggregated by</b>	<a href="#">CompuScales.compuScale</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing <i>CompuMethod</i> in A2L.
compuInverse Value	<a href="#">CompuConst</a>	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se.  <b>Tags:</b> xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale.  <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false







<b>Class</b>	<b>CompuScale</b>			
desc	MultiLanguageOverviewParagraph	0..1	aggr	<p>&lt;desc&gt; represents a general but brief description of the object in question.</p> <p><b>Tags:</b> xml.sequenceOffset=30</p>
lowerLimit	Limit	0..1	attr	<p>This specifies the lower limit of the scale.</p> <p><b>Stereotypes:</b> atpVariation</p> <p><b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=40</p>
mask	PositiveUnlimitedInteger	0..1	attr	<p>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.</p> <p>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.</p> <p>The processing has to be done in order of the COMPU-SCALE elements.</p> <p><b>Tags:</b> xml.sequenceOffset=35</p>
shortLabel	Identifier	0..1	attr	<p>This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier.</p> <p><b>Tags:</b> xml.sequenceOffset=20</p>
symbol	CIdentifier	0..1	attr	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=25</p>
upperLimit	Limit	0..1	attr	<p>This specifies the upper limit of a of the scale.</p> <p><b>Stereotypes:</b> atpVariation</p> <p><b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=50</p>

**Table A.7: CompuScale**

<b>Class</b>	<b>CompuScaleConstantContents</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the fact that a particular scale of the computation method is constant.			
<b>Base</b>	ARObject, CompuScaleContents			
<b>Aggregated by</b>	CompuScale.compuScaleContents			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compuConst	CompuConst	0..1	aggr	<p>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.</p> <p><b>Tags:</b> xml.sequenceOffset=90</p>

**Table A.8: CompuScaleConstantContents**

<b>Class</b>	<b>CompuScales</b>			
<b>Package</b>	M2::MSR::AsamHdo::ComputationMethod			
<b>Note</b>	This meta-class represents the ability to stepwise express a computation method.			
<b>Base</b>	ARObject, CompuContent			
<b>Aggregated by</b>	Compu.compuContent			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=compuScale, compuScale.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

**Table A.9: CompuScales**

<b>Class</b>	<b>CplusplusImplementationDataType</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CplusplusImplementationDataType			
<b>Note</b>	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding			
<b>Base</b>	ARElement, ARObject, AbstractImplementationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, CplusplusImplementationDataTypeContextTarget, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Subclasses</b>	CustomCplusplusImplementationDataType, StdCplusplusImplementationDataType			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CplusplusImplementationDataType has array semantics.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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 CplusplusImplementationDataType.
subElement (ordered)	CplusplusImplementationDataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CplusplusImplementationDataType
template Argument (ordered)	CplusplusTemplateArgument	*	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 CplusplusImplementationDataType is contributed to the language binding.
typeReference	CplusplusImplementationDataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef).

**Table A.10: CplusplusImplementationDataType**

<b>Class</b>	<b>DiagEventDebounceCounterBased</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	<p>This meta-class represents the ability to indicate that the counter-based debounce algorithm shall be used by the DEM for this diagnostic monitor.</p> <p>This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounceCounterBased.</p>			
<b>Base</b>	ARObject, <a href="#">DiagEventDebounceAlgorithm</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticDebounceAlgorithmProps.debounceAlgorithm</a> , <a href="#">DiagnosticEventNeeds.diagEventDebounceAlgorithm</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
counterBasedFdcThresholdStorageValue	Integer	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame.
counterDecrementStepSize	Integer	0..1	attr	<p>This value shall be taken to decrement the internal debounce counter.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterFailedThreshold	Integer	0..1	attr	<p>This value defines the event-specific limit that indicates the "failed" counter status.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterIncrementStepSize	Integer	0..1	attr	<p>This value shall be taken to increment the internal debounce counter.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterJumpDown	Boolean	0..1	attr	<p>This value activates or deactivates the counter jump-down behavior.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterJumpDownValue	Integer	0..1	attr	<p>This value represents the initial value of the internal debounce counter if the counting direction changes from incrementing to decrementing.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterJumpUp	Boolean	0..1	attr	<p>This value activates or deactivates the counter jump-up behavior.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterJumpUpValue	Integer	0..1	attr	<p>This value represents the initial value of the internal debounce counter if the counting direction changes from decrementing to incrementing.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>
counterPassedThreshold	Integer	0..1	attr	<p>This value defines the event-specific limit that indicates the "passed" counter status.</p> <p><b>Stereotypes:</b> atpVariation  <b>Tags:</b> vh.latestBindingTime=preCompileTime</p>

**Table A.11: DiagEventDebounceCounterBased**

<b>Class</b>	<b>DiagEventDebounceMonitorInternal</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>DiagEventDebounceAlgorithm</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Aggregated by</b>	<a href="#">DiagnosticDebounceAlgorithmProps.debounceAlgorithm</a> , <a href="#">DiagnosticEventNeeds.diagEventDebounceAlgorithm</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.12: DiagEventDebounceMonitorInternal**

<b>Class</b>	<b>DiagEventDebounceTimeBased</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	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.			
<b>Base</b>	<i>ARObject</i> , <i>DiagEventDebounceAlgorithm</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Aggregated by</b>	<a href="#">DiagnosticDebounceAlgorithmProps.debounceAlgorithm</a> , <a href="#">DiagnosticEventNeeds.diagEventDebounceAlgorithm</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
timeBasedFdcThresholdStorageValue	TimeValue	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
timeFailedThreshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "failed" status.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
timePassedThreshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "passed" status.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.13: DiagEventDebounceTimeBased**

<b>Class</b>	<b>DiagnosticAbstractDataIdentifier</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class represents an abstract base class for the modeling of a diagnostic data identifier (DID).			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Subclasses</b>	<a href="#">DiagnosticDataIdentifier</a> , <a href="#">DiagnosticDynamicDataIdentifier</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticAbstractDataIdentifier in the scope of diagnostic workflow  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.14: DiagnosticAbstractDataIdentifier**

<b>Class</b>	<b>DiagnosticAbstractParameter</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class represents an abstract base class for modeling a diagnostic parameter.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	DiagnosticParameter, DiagnosticParameterElement			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
bitOffset	PositiveInteger	0..1	attr	This represents the bitOffset of the DiagnosticParameter. The value of the bitOffset shall always be interpreted as relative to the start of the enclosing DiagnosticData Identifier, DiagnosticParameterIdentifier, or Diagnostic RoutineSubfunction.  <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> atp.Status=candidate
dataElement	<a href="#">DiagnosticDataElement</a>	0..1	aggr	This represents the related dataElement of the Diagnostic Parameter  <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=dataElement.shortName, data Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
parameterSize	PositiveInteger	0..1	attr	This attribute allows for the specification of the parameter size. This information is relevant if there is a gap between one diagnostic parameter and the following diagnostic parameter (or the tail of the telegram). The unit is bit and the values shall be multiples of 8.  <b>Tags:</b> atp.Status=candidate

**Table A.15: DiagnosticAbstractParameter**

<b>Class</b>	<b>DiagnosticAccessPermission</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=DiagnosticAccessPermissions			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
authentication Enabled	<a href="#">DiagnosticAuthRole Proxy</a>	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=authenticationEnabled
diagnostic Session	<a href="#">DiagnosticSession</a>	*	ref	This represents the associated DiagnosticSessions  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=diagnosticSession
environmental Condition	<a href="#">Diagnostic EnvironmentalCondition</a>	0..1	ref	This represents the environmental conditions associated with the access permission.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=environmentalCondition





Class	DiagnosticAccessPermission			
securityLevel	<a href="#">DiagnosticSecurityLevel</a>	*	ref	This represents the associated DiagnosticSecurityLevels <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=securityLevel
sovdLock	DiagnosticSovdLock	0..1	ref	This represents the associated SOVD lock. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=sovdLock atp.Status=candidate

**Table A.16: DiagnosticAccessPermission**

Class	DiagnosticAging			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticAging			
<b>Note</b>	Defines the aging algorithm. <b>Tags:</b> atp.recommendedPackage=DiagnosticAgings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
agingCycle	<a href="#">DiagnosticOperation Cycle</a>	0..1	ref	This represents the applicable aging cycle. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=agingCycle.diagnosticOperationCycle, aging Cycle.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
threshold	PositiveInteger	0..1	attr	Number of aging cycles needed to unlearn/delete the event. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.17: DiagnosticAging**

Class	DiagnosticAuthRole			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	This meta-class represents the ability to specify an authentication role that can be used to deliver fine-grained access rights. <b>Tags:</b> atp.recommendedPackage=DiagnosticAuthRoles			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">Multilanguage</a> <a href="#">Referrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	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.18: DiagnosticAuthRole**

<b>Class</b>	<b>DiagnosticAuthRoleProxy</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	<a href="#">DiagnosticAccessPermission.authenticationEnabled</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
authentication Role	<a href="#">DiagnosticAuthRole</a>	*	ref	This reference identifies the authenticationRole applicable for the enclosing DiagnosticAccessPermission.

**Table A.19: DiagnosticAuthRoleProxy**

<b>Class</b>	<b>DiagnosticAuthentication</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::Authentication			
<b>Note</b>	This meta-class represents the ability to configure the usage of the UDS service Authentication in the Diagnostic extract.			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	DiagnosticAuthTransmitCertificate, DiagnosticAuthenticationConfiguration, DiagnosticDeAuthentication, DiagnosticProofOfOwnership, DiagnosticVerifyCertificateBidirectional, DiagnosticVerifyCertificateUnidirectional			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.20: DiagnosticAuthentication**

<b>Class</b>	<b>DiagnosticClearCondition</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
<b>Note</b>	This meta-class describes a clear condition for diagnostic purposes. <b>Tags:</b> atp.recommendedPackage=DiagnosticConditions			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticCondition</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.21: DiagnosticClearCondition**

<b>Enumeration</b>	<b>DiagnosticClearDtcLimitationEnum</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
<b>Note</b>	Scope of the DEM_ClearDTC Api.			
<b>Aggregated by</b>	<a href="#">DiagnosticMemoryDestination.clearDtcLimitation</a>			





Enumeration	DiagnosticClearDtcLimitationEnum
Literal	Description
allSupportedDtcs	DEM_ClearDtc API accepts all supported DTC values. <b>Tags:</b> atp.EnumerationLiteralIndex=0
clearAllDtcs	DEM_ClearDtc API accepts ClearAllDTCs only. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.22: DiagnosticClearDtcLimitationEnum**

Enumeration	DiagnosticClearEventAllowedBehaviorEnum
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent
<b>Note</b>	This enumeration defines the possible behavior for clear event allowed
<b>Aggregated by</b>	<a href="#">DiagnosticEvent.clearEventAllowedBehavior</a>
Literal	Description
noStatusByteChange	The event status byte keeps unchanged. <b>Tags:</b> atp.EnumerationLiteralIndex=0
onlyThisCycleAndReadiness	The OperationCycle and readiness bits of the event status byte are reset. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.23: DiagnosticClearEventAllowedBehaviorEnum**

Class	DiagnosticComControl			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl			
<b>Note</b>	This represents an instance of the "Communication Control" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticCommunicationControls			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	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.24: DiagnosticComControl**

Class	<<atpVariation>> DiagnosticCommonProps
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
<b>Note</b>	This meta-class aggregates a number of common properties that are shared among a diagnostic extract. <b>Tags:</b> vh.latestBindingTime=codeGenerationTime
<b>Base</b>	<a href="#">ARObject</a>
<b>Aggregated by</b>	<a href="#">DiagnosticContributionSet.commonProperties</a>







Class	<<atpVariation>> DiagnosticCommonProps			
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	<a href="#">DiagnosticDebounce AlgorithmProps</a>	*	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..*.
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	<a href="#">SoftwareCluster DiagnosticAddress</a>	*	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). <b>Stereotypes:</b> atpSplitable <b>Tags:</b> xml.namePlural=DIAGNOSTIC-ADDRESSES
event Combination Reporting Behavior	<a href="#">DiagnosticEvent CombinationReporting BehaviorEnum</a>	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). <b>Stereotypes:</b> atpSplitable
maxNumberOf Request Correctly Received Response Pending	PositiveInteger	0..1	attr	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceived-ResponsePending) allowed per request. DCM will send a negative response with response code 0x10 (generalReject), in case the limit value gets reached. Value 0xFF means that no limit number of NRC 0x78 response apply.
occurrence Counter Processing	<a href="#">DiagnosticOccurrence CounterProcessing Enum</a>	0..1	attr	This attribute defines the consideration of the fault confirmation process for the occurrence counter.
resetConfirmed BitOnOverflow	Boolean	0..1	attr	This attribute defines, whether the confirmed bit is reset or not while an event memory entry will be displaced.
resetPendingBit OnOverflow	Boolean	0..1	attr	This attribute defines, whether the pending bit is reset or not while an event memory entry will be displaced. In order to be compliant to ISO 14229-1 [1], this parameter needs to be set to "false".
responseOnAll RequestSids	Boolean	0..1	attr	If set to FALSE the DCM will not respond to diagnostic request that contains a service ID which is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF (Response IDs).
responseOn Second Declined Request	Boolean	0..1	attr	Defines the reaction upon a second request (ClientB) that can not be processed (e.g. due to priority assessment). TRUE: when the second request (Client B) can not be processed, it shall be answered with NRC21 BusyRepeat Request. FALSE: when the second request (Client B) can not be processed, it shall not be responded.





<b>Class</b>	<<atpVariation>> <b>DiagnosticCommonProps</b>			
typeOfEventCombinationSupported	<a href="#">DiagnosticEventCombinationBehaviorEnum</a>	0..1	attr	Select type of Event Combination support.

**Table A.25: DiagnosticCommonProps**

<b>Class</b>	<b>DiagnosticCommunicationManagerNeeds</b>			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
<b>Note</b>	Specifies the general needs on the configuration of the Diagnostic Communication Manager (Dcm) which are not related to a particular item (e.g. a PID or DiagnosticRoutineNeeds). The main use case is the mapping of service ports to the Dcm which are not related to a particular item.			
<b>Base</b>	ARObject, DiagnosticCapabilityElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a> , <a href="#">ServiceNeeds</a>			
<b>Aggregated by</b>	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
serviceRequestCallbackType	DiagnosticServiceRequestCallbackTypeEnum	0..1	attr	This represents the ability to define whether the usage of PortInterface ServiceRequestNotification has the characteristics of being initiated by a manufacturer or by a supplier.

**Table A.26: DiagnosticCommunicationManagerNeeds**

<b>Class</b>	<b>DiagnosticCondition</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
<b>Note</b>	Abstract element for StorageConditions and EnableConditions.			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DiagnosticClearCondition</a> , <a href="#">DiagnosticEnableCondition</a> , <a href="#">DiagnosticStorageCondition</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.27: DiagnosticCondition**

<b>Class</b>	<b>DiagnosticConditionInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform.  <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.28: DiagnosticConditionInterface**

<b>Class</b>	<b>DiagnosticConnectedIndicator</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
<b>Note</b>	Description of indicators that are defined per DiagnosticEvent.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticEvent.connectedIndicator</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
behavior	DiagnosticConnectedIndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycle	<a href="#">DiagnosticOperationCycle</a>	0..1	ref	The deactivation of indicators per event is defined as healing of a diagnostic event. The operation cycle in which the warning indicator will be switched off is defined here.
healingCycleCounterThreshold	PositiveInteger	0..1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
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.29: DiagnosticConnectedIndicator**

<b>Class</b>	<b>DiagnosticContributionSet</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=DiagnosticContributionSets			
<b>Base</b>	ARElement, ARObject, CollectableElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
commonProperties	<a href="#">DiagnosticCommonProps</a>	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire DiagnosticContributionSet. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=commonProperties
element	DiagnosticCommonElement	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> 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. <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=serviceTable.diagnosticServiceTable, serviceTable.variationPoint.shortLabel, vh.latestBindingTime=postBuild

**Table A.30: DiagnosticContributionSet**

<b>Class</b>	<b>DiagnosticControlDTCSetting</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ControlDTCSetting			
<b>Note</b>	This represents an instance of the "Control DTC Setting" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticControlDtcSettings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dtcSettingClass	DiagnosticControlIDTCSettingClass	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 DiagnosticControlIDTCSetting in the given context.

**Table A.31: DiagnosticControlDTCSetting**

<b>Class</b>	<b>DiagnosticCustomServiceInstance</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance			
<b>Note</b>	This meta-class has the ability to define an instance of a custom diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticCustomInstances			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
customServiceClass	DiagnosticCustomServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustomServiceClass.

**Table A.32: DiagnosticCustomServiceInstance**

<b>Class</b>	<b>DiagnosticDTCInformationInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.33: DiagnosticDTCInformationInterface**

<b>Class</b>	<b>DiagnosticDataByIdentifier</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
<b>Note</b>	This represents an abstract base class for all diagnostic services that access data by identifier.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			





<b>Class</b>	<i>DiagnosticDataByIdentifier</i> (abstract)			
<b>Subclasses</b>	<a href="#">DiagnosticReadDataByIdentifier</a> , <a href="#">DiagnosticReadScalingDataByIdentifier</a> , <a href="#">DiagnosticWriteDataByIdentifier</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataIdentifier	<a href="#">DiagnosticAbstractDataIdentifier</a>	0..1	ref	This represents the linked DiagnosticDataIdentifier.

**Table A.34: DiagnosticDataByIdentifier**

<b>Class</b>	<b>DiagnosticDataElement</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
<b>Base</b>	<i>ARObject</i> , <i>DiagnosticServiceMappingDiagTarget</i> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticAbstractParameter.dataElement</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticDataElement if used with <a href="#">DiagnosticReadScalingDataByIdentifier</a>
swDataDef Props	<a href="#">SwDataDefProps</a>	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.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=swDataDefProps

**Table A.35: DiagnosticDataElement**

<b>Class</b>	<b>DiagnosticDataElementInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform.  <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , <i>DiagnosticAbstractDataIdentifierInterface</i> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of an element of a diagnostic data identifier.

**Table A.36: DiagnosticDataElementInterface**

<b>Class</b>	<b>DiagnosticDataIdentifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. <b>Tags:</b> atp.recommendedPackage=DiagnosticDataIdentifiers			
<b>Base</b>	ARElement, ARObject, CollectableElement, <a href="#">DiagnosticAbstractDataIdentifier</a> , DiagnosticCommonElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">DiagnosticParameter</a>	*	aggr	This is the dataElement associated with the Diagnostic DataIdentifier. <b>Stereotypes:</b> atp.Splittable; atp.Variation <b>Tags:</b> 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 Diagnostic DataIdentifier.
representsVin	Boolean	0..1	attr	This attribute indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfoByte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

**Table A.37: DiagnosticDataIdentifier**

<b>Class</b>	<b>DiagnosticDataIdentifierGenericInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticAbstractDataIdentifierInterface</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.38: DiagnosticDataIdentifierGenericInterface**

<b>Class</b>	<b>DiagnosticDataIdentifierInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticAbstractDataIdentifierInterface</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





Class		DiagnosticDataIdentifierInterface		
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.39: DiagnosticDataIdentifierInterface**

Class		DiagnosticDataIdentifierSet		
Package		M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode		
Note		This represents the ability to define a list of DiagnosticDataIdentifiers that can be reused in different contexts. <b>Tags:</b> atp.recommendedPackage=DiagnosticDataIdentifierSets		
Base		<i>ARElement, ARObjct, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>		
Aggregated by		ARPackage.element		
Attribute	Type	Mult.	Kind	Note
dataidentifier (ordered)	<a href="#">DiagnosticDataIdentifier</a>	*	ref	Reference to an ordered list of Data Identifiers.

**Table A.40: DiagnosticDataIdentifierSet**

Class		DiagnosticDataPortMapping		
Package		M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping		
Note		This meta-class provides the ability to define a diagnostic access to an entire DID. <b>Tags:</b> atp.recommendedPackage=DiagnosticServiceMappings		
Base		<i>ARElement, ARObjct, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>		
Aggregated by		ARPackage.element		
Attribute	Type	Mult.	Kind	Note
diagnosticData Element	<a href="#">DiagnosticDataElement</a>	0..1	ref	This reference represents the applicable DiagnosticData Element.
diagnosticData Identifier	<a href="#">DiagnosticDataIdentifier</a>	0..1	ref	This reference represents the applicable DiagnosticData Identifier.
pPortPrototype InExecutable	<a href="#">PPortPrototype</a>	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). <b>Stereotypes:</b> atpUriDef <b>InstanceRef implemented by:</b> 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. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=process

**Table A.41: DiagnosticDataPortMapping**

<b>Class</b>	<b>DiagnosticDebounceAlgorithmProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticDebouncingAlgorithm			
<b>Note</b>	Defines properties for the debounce algorithm class.			
<b>Base</b>	<i>ARObject</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Aggregated by</b>	<a href="#">DiagnosticCommonProps.debounceAlgorithmProps</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
debounce Algorithm	DiagEventDebounce Algorithm	0..1	aggr	This represents the actual debounce algorithm.
debounce Behavior	<a href="#">DiagnosticDebounce BehaviorEnum</a>	0..1	attr	This attribute defines how the event debounce algorithm will behave, if a related enable condition is not fulfilled or ControlDTCSetting of the related event is disabled. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.42: DiagnosticDebounceAlgorithmProps**

<b>Enumeration</b>	<b>DiagnosticDebounceBehaviorEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticDebouncingAlgorithm
<b>Note</b>	Event debounce algorithm behavior options.
<b>Aggregated by</b>	<a href="#">DiagnosticDebounceAlgorithmProps.debounceBehavior</a>
<b>Literal</b>	<b>Description</b>
freeze	The event debounce counter will be frozen with the current value and will not change while a related enable condition is not fulfilled or ControlDTCSetting of the related event is disabled. After all related enable conditions are fulfilled and ControlDTCSetting of the related event is enabled again, the event qualification will continue with the next report of the event (i.e. SetEventStatus). <b>Tags:</b> atp.EnumerationLiteralIndex=0
reset	The event debounce counter will be reset to initial value if a related enable condition is not fulfilled or ControlDTCSetting of the related event is disabled. The qualification of the event will be restarted with the next valid event report. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.43: DiagnosticDebounceBehaviorEnum**

<b>Class</b>	<b>DiagnosticDoIPActivationLineInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , <i>DiagnosticPortInterface</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>PortInterface</i> , <i>Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.44: DiagnosticDoIPActivationLineInterface**



<b>Class</b>	<b>DiagnosticDoIPEntityIdentificationInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement the DoIP Entity Identification on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.45: DiagnosticDoIPEntityIdentificationInterface**

<b>Class</b>	<b>DiagnosticDoIPGroupIdentificationInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.46: DiagnosticDoIPGroupIdentificationInterface**

<b>Class</b>	<b>DiagnosticDoIPPowerModelInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.47: DiagnosticDoIPPowerModelInterface**

<b>Class</b>	<b>DiagnosticDoIPTriggerVehicleAnnouncementInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			





<b>Class</b>	<b>DiagnosticDolPTriggerVehicleAnnouncementInterface</b>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.48: DiagnosticDolPTriggerVehicleAnnouncementInterface**

<b>Class</b>	<b>DiagnosticDownloadInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.49: DiagnosticDownloadInterface**

<b>Class</b>	<b>DiagnosticDynamicallyDefineDataIdentifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
<b>Note</b>	This represents an instance of the "Dynamically Define Data Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataIdentifier	DiagnosticDynamicData Identifier	0..1	ref	This represents the applicable DiagnosticDynamicData Identifier.
dynamically DefineData IdentifierClass	<a href="#">DiagnosticDynamically DefineDataIdentifier Class</a>	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.50: DiagnosticDynamicallyDefineDataIdentifier**

<b>Class</b>	<b>DiagnosticDynamicallyDefineDataIdentifierClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Dynamically Define Data Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			





<b>Class</b>		<b>DiagnosticDynamicallyDefineDataIdentifierClass</b>		
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	DiagnosticHandleDDDIDConfigurationEnum	0..1	attr	This configuration switch defines whether DDDID 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.51: DiagnosticDynamicallyDefineDataIdentifierClass**

<b>Class</b>		<b>DiagnosticEcuReset</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
<b>Note</b>	This represents an instance of the "ECU Reset" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticEcuResets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	<a href="#">DiagnosticEcuResetClass</a>	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.52: DiagnosticEcuReset**

<b>Class</b>		<b>DiagnosticEcuResetClass</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Ecu Reset" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticEcuResets			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
respondToReset	<a href="#">DiagnosticResponseToEcuResetEnum</a>	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.53: DiagnosticEcuResetClass**

<b>Class</b>	<b>DiagnosticEnableCondition</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
<b>Note</b>	Specification of an enable condition. <b>Tags:</b> atp.recommendedPackage=DiagnosticConditions			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticCondition</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , PackageableElement, <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.54: DiagnosticEnableCondition**

<b>Class</b>	<b>DiagnosticEnableConditionPortMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Defines to which SWC service ports the DiagnosticEnableCondition is mapped. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , PackageableElement, <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
enableCondition	<a href="#">DiagnosticEnableCondition</a>	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=process
rPortPrototypeInExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the DiagnosticEnableConditionPortMapping on the AUTOSAR adaptive platform. <b>Stereotypes:</b> atpUriDef <b>InstanceRef implemented by:</b> RPortPrototypeInExecutableInstanceRef

**Table A.55: DiagnosticEnableConditionPortMapping**

<b>Class</b>	<b>DiagnosticEnvCompareCondition</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">DiagnosticEnvConditionFormulaPart</a>			
<b>Subclasses</b>	<a href="#">DiagnosticEnvDataCondition</a> , DiagnosticEnvDataElementCondition, DiagnosticEnvModeCondition			
<b>Aggregated by</b>	<a href="#">DiagnosticEnvConditionFormula.part</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compareType	DiagnosticCompareTypeEnum	0..1	attr	This attributes represents the concrete type of the comparison.

**Table A.56: DiagnosticEnvCompareCondition**

<b>Class</b>	<b>DiagnosticEnvConditionFormula</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
<b>Note</b>	<p>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.</p> <p>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.</p>			
<b>Base</b>	ARObject, <a href="#">DiagnosticEnvConditionFormulaPart</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticEnvConditionFormula.part</a> , <a href="#">DiagnosticEnvironmentalCondition.formula</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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)	<a href="#">DiagnosticEnvConditionFormulaPart</a>	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

**Table A.57: DiagnosticEnvConditionFormula**

<b>Class</b>	<b>DiagnosticEnvConditionFormulaPart</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
<b>Note</b>	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompareCondition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
<b>Base</b>	ARObject			
<b>Subclasses</b>	<a href="#">DiagnosticEnvCompareCondition</a> , <a href="#">DiagnosticEnvConditionFormula</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticEnvConditionFormula.part</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.58: DiagnosticEnvConditionFormulaPart**

<b>Class</b>	<b>DiagnosticEnvDataCondition</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">DiagnosticEnvCompareCondition</a> , <a href="#">DiagnosticEnvConditionFormulaPart</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticEnvConditionFormula.part</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
compareValue	ValueSpecification	0..1	aggr	This attribute represents a fixed compare value taken to evaluate the compare condition.
dataElement	<a href="#">DiagnosticDataElement</a>	0..1	ref	This reference represents the related diagnostic data element.

**Table A.59: DiagnosticEnvDataCondition**

<b>Class</b>	<b>DiagnosticEnvironmentalCondition</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
<b>Note</b>	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). <b>Tags:</b> atp.recommendedPackage=DiagnosticEnvironmentalConditions			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
formula	DiagnosticEnvCondition Formula	0..1	aggr	This attribute represents the formula part of the DiagnosticEnvironmentalCondition.
modeElement	DiagnosticEnvMode Element	*	aggr	This aggregation contains a representation of Mode Declarations in the context of a DiagnosticEnvironmentalCondition.

**Table A.60: DiagnosticEnvironmentalCondition**

<b>Class</b>	<b>DiagnosticEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
<b>Note</b>	This element is used to configure DiagnosticEvents. <b>Tags:</b> atp.recommendedPackage=DiagnosticEvents			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators.  <b>Stereotypes:</b> atpSplitable; atpVariation <b>Tags:</b> atp.Splitkey=connectedIndicator.shortName, connectedIndicator.variationPoint.shortLabel vh.latestBindingTime=postBuild





<b>Class</b>	<b>DiagnosticEvent</b>			
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.61: DiagnosticEvent**

<b>Enumeration</b>	<b>DiagnosticEventCombinationBehaviorEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
<b>Note</b>	Select type of Event Combination support
<b>Aggregated by</b>	<a href="#">DiagnosticCommonProps.typeOfEventCombinationSupported</a>
<b>Literal</b>	<b>Description</b>
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. <b>Tags:</b> 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. <b>Tags:</b> atp.EnumerationLiteralIndex=0

**Table A.62: DiagnosticEventCombinationBehaviorEnum**

<b>Enumeration</b>	<b>DiagnosticEventCombinationReportingBehaviorEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
<b>Note</b>	Select reporting format of events. Applicable only for Event Combination on Retrieval.
<b>Aggregated by</b>	<a href="#">DiagnosticCommonProps.eventCombinationReportingBehavior</a>
<b>Literal</b>	<b>Description</b>
reportingIn ChronologicalOrder OldestFirst	The reporting order for event combination on retrieval is the chronological storage order of the events <b>Tags:</b> atp.EnumerationLiteralIndex=0

**Table A.63: DiagnosticEventCombinationReportingBehaviorEnum**

<b>Enumeration</b>	<b>DiagnosticEventDisplacementStrategyEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
<b>Note</b>	Defines the displacement strategy.
<b>Aggregated by</b>	<a href="#">DiagnosticMemoryDestination.eventDisplacementStrategy</a>





<b>Enumeration</b>	<b>DiagnosticEventDisplacementStrategyEnum</b>
<b>Literal</b>	<b>Description</b>
full	Event memory entry displacement is enabled, by consideration of priority active/passive status, and occurrence. <b>Tags:</b> atp.EnumerationLiteralIndex=0
none	Event memory entry displacement is disabled. <b>Tags:</b> atp.EnumerationLiteralIndex=1
prioOcc	Event memory entry displacement is enabled, by consideration of priority and occurrence (but without active/passive status). <b>Tags:</b> atp.EnumerationLiteralIndex=2

**Table A.64: DiagnosticEventDisplacementStrategyEnum**

<b>Class</b>	<b>DiagnosticEventInterface</b>				
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface				
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces				
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>				
<b>Aggregated by</b>	ARPackage.element				
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>	
–	–	–	–	–	

**Table A.65: DiagnosticEventInterface**

<b>Class</b>	<b>DiagnosticEventPortMapping</b>				
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping				
<b>Note</b>	Defines to which SWC service ports the DiagnosticEvent is mapped. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings				
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>				
<b>Aggregated by</b>	ARPackage.element				
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>	
diagnosticEvent	<a href="#">DiagnosticEvent</a>	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.  <b>Stereotypes:</b> atp.Splittable <b>Tags:</b> atp.Splitkey=process	
rPortPrototype InExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the DiagnosticEventPortMapping on the AUTOSAR adaptive platform.  <b>Stereotypes:</b> atp.UriDef <b>InstanceRef implemented by:</b> RPortPrototypeInExecutableInstanceRef	

**Table A.66: DiagnosticEventPortMapping**



<b>Class</b>	<b>DiagnosticEventToDebounceAlgorithmMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Defines which Debounce Algorithm is applicable for a DiagnosticEvent. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
debounceAlgorithm	<a href="#">DiagnosticDebounceAlgorithmProps</a>	0..1	ref	Reference to a DebounceAlgorithm assigned to a DiagnosticEvent.
diagnosticEvent	<a href="#">DiagnosticEvent</a>	0..1	ref	Reference to a DiagnosticEvent to which a Debounce Algorithm is assigned.

**Table A.67: DiagnosticEventToDebounceAlgorithmMapping**

<b>Class</b>	<b>DiagnosticEventToEnableConditionGroupMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Defines which EnableConditionGroup is applicable for a DiagnosticEvent. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticEvent	<a href="#">DiagnosticEvent</a>	0..1	ref	Reference to a DiagnosticEvent to which an Enable ConditionGroup is assigned.
enableConditionGroup	DiagnosticEnableConditionGroup	0..1	ref	Reference to an EnableConditionGroup assigned to a DiagnosticEvent.

**Table A.68: DiagnosticEventToEnableConditionGroupMapping**

<b>Class</b>	<b>DiagnosticEventToOperationCycleMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Defines which OperationCycle is applicable for a DiagnosticEvent. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticEvent	<a href="#">DiagnosticEvent</a>	0..1	ref	Reference to a DiagnosticEvent to which an Operation Cycle is assigned.
operationCycle	<a href="#">DiagnosticOperationCycle</a>	0..1	ref	Reference to an OperationCycle assigned to a Diagnostic Event.

**Table A.69: DiagnosticEventToOperationCycleMapping**

<b>Class</b>	<b>DiagnosticEventToTroubleCodeUdsMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Defines which UDS Diagnostic Trouble Code is applicable for a DiagnosticEvent. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticEvent	<a href="#">DiagnosticEvent</a>	0..1	ref	Reference to a DiagnosticEvent to which a UDS Diagnostic Trouble Code is assigned.
troubleCodeUds	<a href="#">DiagnosticTroubleCodeUds</a>	0..1	ref	Reference to an UDS Diagnostic Trouble Code assigned to a DiagnosticEvent.

**Table A.70: DiagnosticEventToTroubleCodeUdsMapping**

<b>Class</b>	<b>DiagnosticEventWindow</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
<b>Note</b>	This represents the ability to define the characteristics of the applicable event window			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	<a href="#">DiagnosticResponseOnEvent.eventWindow</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventWindow Time	DiagnosticEventWindow TimeEnum	0..1	attr	This attribute clarifies the validity of the eventWindow

**Table A.71: DiagnosticEventWindow**

<b>Class</b>	<b>DiagnosticExtendedDataRecord</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticExtendedDataRecord			
<b>Note</b>	Description of an extended data record. <b>Tags:</b> atp.recommendedPackage=DiagnosticExtendedDataRecords			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
recordElement	<a href="#">DiagnosticParameter</a>	*	aggr	Defined DataElements in the extended record element. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=recordElement.bitOffset, recordElement.ident.shortName
recordNumber	PositiveInteger	0..1	attr	This attribute specifies an unique identifier for an extended data record.
trigger	<a href="#">DiagnosticRecordTriggerEnum</a>	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.72: DiagnosticExtendedDataRecord**

<b>Class</b>	<b>DiagnosticFreezeFrame</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticFreezeFrame			
<b>Note</b>	This element describes combinations of DIDs for a non OBD relevant freeze frame. <b>Tags:</b> atp.recommendedPackage=DiagnosticFreezeFrames			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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.73: DiagnosticFreezeFrame**

<b>Class</b>	<b>DiagnosticGenericUdsInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.74: DiagnosticGenericUdsInterface**

<b>Class</b>	<b>DiagnosticIndicatorInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.75: DiagnosticIndicatorInterface**

<b>Class</b>	<b>DiagnosticMapping</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
<b>Note</b>	Abstract element for different kinds of diagnostic mappings.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DiagnosticEventToDebounceAlgorithmMapping</a> , <a href="#">DiagnosticEventToEnableConditionGroupMapping</a> , <a href="#">DiagnosticEventToOperationCycleMapping</a> , <a href="#">DiagnosticEventToSecurityEventMapping</a> , <a href="#">DiagnosticEventToTroubleCodeUdsMapping</a> , <a href="#">DiagnosticFimAliasEventGroupMapping</a> , <a href="#">DiagnosticFimAliasEventMapping</a> , <a href="#">DiagnosticInhibitSourceEventMapping</a> , <a href="#">DiagnosticMasterToSlaveEventMapping</a> , <a href="#">DiagnosticProvidedDataMapping</a> , <a href="#">DiagnosticSecureCodingMapping</a> , <a href="#">DiagnosticSovdConfigurationDataIdentifierMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">DiagnosticTroubleCodeUdsToClearConditionGroupMapping</a> , <a href="#">DiagnosticTroubleCodeUdsToTroubleCodeObdMapping</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.76: DiagnosticMapping**

<b>Class</b>	<b>DiagnosticMemoryDestination</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
<b>Note</b>	This abstract meta-class represents a possible memory destination for a diagnostic event.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DiagnosticMemoryDestinationPrimary</a> , <a href="#">DiagnosticMemoryDestinationUserDefined</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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	<a href="#">DiagnosticClearDtcLimitationEnum</a>	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	<a href="#">DiagnosticEventDisplacementStrategyEnum</a>	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	<a href="#">DiagnosticMemoryEntryStorageTriggerEnum</a>	0..1	attr	Describes the trigger to allocate an event memory entry.
statusBitHandlingTestFailedSinceLastClear	<a href="#">DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum</a>	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.





Class	<i>DiagnosticMemoryDestination</i> (abstract)			
statusBitStorageTestFailed	Boolean	0..1	attr	This parameter is used to activate/deactivate the permanent storage of the "TestFailed" status bits. true: storage activated false: storage deactivated On the adaptive platform, this attribute shall always be set to true.
typeOfFreezeFrameRecordNumeration	<a href="#">DiagnosticTypeOfFreezeFrameRecordNumerationEnum</a>	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

**Table A.77: DiagnosticMemoryDestination**

Class	<i>DiagnosticMemoryDestinationPortMapping</i>			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticMemoryDestination. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
memoryDestination	<a href="#">DiagnosticMemoryDestination</a>	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. <b>Stereotypes:</b> atp.Splitable <b>Tags:</b> atp.Splitkey=process
rPortPrototypeInExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the DiagnosticMemoryDestinationMapping on the AUTOSAR adaptive platform. <b>Stereotypes:</b> atp.UriDef <b>InstanceRef implemented by:</b> RPortPrototypeInExecutableInstanceRef

**Table A.78: DiagnosticMemoryDestinationPortMapping**

Class	<i>DiagnosticMemoryDestinationPrimary</i>			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
Note	This represents a primary memory for a diagnostic event. <b>Tags:</b> atp.recommendedPackage=DiagnosticMemoryDestinations			
Base	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMemoryDestination</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
typeOfDtcSupported	DiagnosticTypeOfDtcSupportedEnum	0..1	attr	This attribute defines the format returned by Dem_DcmGetTranslationType and does not relate to/influence the supported Dem functionality.

**Table A.79: DiagnosticMemoryDestinationPrimary**

<b>Class</b>	<b>DiagnosticMemoryDestinationUserDefined</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
<b>Note</b>	This represents a user-defined memory for a diagnostic event. <b>Tags:</b> atp.recommendedPackage=DiagnosticMemoryDestinations			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMemoryDestination</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
authRole	<a href="#">DiagnosticAuthRole</a>	*	ref	This reference identifies the collection of applicable DiagnosticAuthRole <b>Stereotypes:</b> atp.Splittable <b>Tags:</b> atp.Splitkey=authRole
memoryId	PositiveInteger	0..1	attr	This represents the identifier of the user-defined memory.

**Table A.80: DiagnosticMemoryDestinationUserDefined**

<b>Class</b>	<b>DiagnosticMonitorInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.81: DiagnosticMonitorInterface**

<b>Class</b>	<b>DiagnosticMultipleConditionPortMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
<b>Note</b>	Defines to which SWC service port that can handle a collection of diagnostic conditions the specific condition is mapped. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticMultipleResourcePortMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticCondition	<a href="#">DiagnosticCondition</a>	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. <b>Stereotypes:</b> atp.Splittable <b>Tags:</b> atp.Splitkey=process





<b>Class</b>	<b>DiagnosticMultipleConditionPortMapping</b>			
rPortPrototype InExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the Diagnostic ConditionPortMapping on the AUTOSAR adaptive platform.  <b>Stereotypes:</b> atpUriDef <b>InstanceRef implemented by:</b> RPortPrototypeln ExecutableInstanceRef

**Table A.82: DiagnosticMultipleConditionPortMapping**

<b>Class</b>	<b>DiagnosticMultipleEventPortMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
<b>Note</b>	Defines to which SWC service port that can handle a collection of event status the specific event is mapped. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortMappings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticMultipleResourcePortMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticEvent	<a href="#">DiagnosticEvent</a>	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.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=process
rPortPrototype InExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleEventPortMapping on the AUTOSAR adaptive platform.  <b>Stereotypes:</b> atpUriDef <b>InstanceRef implemented by:</b> RPortPrototypeln ExecutableInstanceRef

**Table A.83: DiagnosticMultipleEventPortMapping**

<b>Class</b>	<b>DiagnosticMultipleMonitorPortMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
<b>Note</b>	Defines to which SWC service port that can handle a collection of monitors the specific event is mapped <b>Tags:</b> atp.recommendedPackage=DiagnosticPortMappings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticMultipleResourcePortMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticEvent	<a href="#">DiagnosticEvent</a>	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port.





<b>Class</b>		<b>DiagnosticMultipleMonitorPortMapping</b>		
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. <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=process
rPortPrototype InExecutable	<a href="#">RPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleMonitorPortMapping on the AUTOSAR adaptive platform. <b>Stereotypes:</b> atpUriDef <b>InstanceRef implemented by:</b> RPortPrototypeInExecutableInstanceRef

**Table A.84: DiagnosticMultipleMonitorPortMapping**

<b>Class</b>	<b>DiagnosticMultipleResourcePortMapping</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
<b>Note</b>	This abstract base class enables the mapping of diagnostic PortInterfaces that deal with multiple diagnostic resources.			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DiagnosticMultipleConditionPortMapping</a> , <a href="#">DiagnosticMultipleEventPortMapping</a> , <a href="#">DiagnosticMultipleMonitorPortMapping</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.85: DiagnosticMultipleResourcePortMapping**

<b>Enumeration</b>	<b>DiagnosticOccurrenceCounterProcessingEnum</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps		
<b>Note</b>	The occurrence counter triggering types.		
<b>Aggregated by</b>	<a href="#">DiagnosticCommonProps.occurrenceCounterProcessing</a>		
<b>Literal</b>	<b>Description</b>		
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). <b>Tags:</b> atp.EnumerationLiteralIndex=0		
testFailedBit	The occurrence counter is incremented when TestFailed bit transitions from 0 to 1 (and the fault confirmation is not considered). <b>Tags:</b> atp.EnumerationLiteralIndex=1		

**Table A.86: DiagnosticOccurrenceCounterProcessingEnum**

<b>Class</b>	<b>DiagnosticOperationCycle</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle		
<b>Note</b>	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. <b>Tags:</b> atp.recommendedPackage=DiagnosticOperationCycles		







<b>Class</b>	<b>DiagnosticOperationCycle</b>			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
type	DiagnosticOperationCycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

**Table A.87: DiagnosticOperationCycle**

<b>Class</b>	<b>DiagnosticOperationCycleInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.88: DiagnosticOperationCycleInterface**

<b>Class</b>	<b>DiagnosticParameter</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	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.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticAbstractParameter</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticDataIdentifier.dataElement</a> , <a href="#">DiagnosticExtendedDataRecord.recordElement</a> , <a href="#">DiagnosticInfoType.dataElement</a> , <a href="#">DiagnosticParameterIdentifier.dataElement</a> , <a href="#">DiagnosticRequestRoutineResults.request</a> , <a href="#">DiagnosticRequestRoutineResults.response</a> , <a href="#">DiagnosticStartRoutine.request</a> , <a href="#">DiagnosticStartRoutine.response</a> , <a href="#">DiagnosticStopRoutine.request</a> , <a href="#">DiagnosticStopRoutine.response</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
ident	DiagnosticParameterIdent	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).  <b>Stereotypes:</b> atpIdentityContributor
supportInfo	DiagnosticParameterSupportInfo	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.89: DiagnosticParameter**

<b>Class</b>	<b>DiagnosticPeriodicRate</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
<b>Note</b>	This represents the ability to define a periodic rate for the specification of the "read data by periodic ID" diagnostic service.			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	<a href="#">DiagnosticReadDataByPeriodicIDClass.periodicRate</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.90: DiagnosticPeriodicRate**

<b>Class</b>	<b>DiagnosticPortInterface</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces.			
<b>Base</b>	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
<b>Subclasses</b>	<a href="#">DiagnosticAbstractDataIdentifierInterface</a> , <a href="#">DiagnosticAbstractRoutineInterface</a> , <a href="#">DiagnosticAuthenticationInterface</a> , <a href="#">DiagnosticComControllInterface</a> , <a href="#">DiagnosticConditionInterface</a> , <a href="#">DiagnosticDTCInformationInterface</a> , <a href="#">DiagnosticDoIPActivationLineInterface</a> , <a href="#">DiagnosticDoIPEntityIdentificationInterface</a> , <a href="#">DiagnosticDoIPGroupIdentificationInterface</a> , <a href="#">DiagnosticDoIPPowerModeInterface</a> , <a href="#">DiagnosticDoIPTriggerVehicleAnnouncementInterface</a> , <a href="#">DiagnosticDownloadInterface</a> , <a href="#">DiagnosticEcuResetInterface</a> , <a href="#">DiagnosticEventInterface</a> , <a href="#">DiagnosticExternalAuthenticationInterface</a> , <a href="#">DiagnosticGenericUdsInterface</a> , <a href="#">DiagnosticIndicatorInterface</a> , <a href="#">DiagnosticMonitorInterface</a> , <a href="#">DiagnosticMultipleResourceInterface</a> , <a href="#">DiagnosticOperationCycleInterface</a> , <a href="#">DiagnosticRequestFileTransferInterface</a> , <a href="#">DiagnosticSecurityLevelInterface</a> , <a href="#">DiagnosticServiceValidationInterface</a> , <a href="#">DiagnosticSovdPortInterface</a> , <a href="#">DiagnosticUploadInterface</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.91: DiagnosticPortInterface**

<b>Class</b>	<b>DiagnosticProvidedDataMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
<b>Note</b>	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. <b>Tags:</b> atp.recommendedPackage=DataMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dataElement	<a href="#">DiagnosticDataElement</a>	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider.
dataProvider	NameToken	0..1	attr	This represents the ability to further specify the data provider.

**Table A.92: DiagnosticProvidedDataMapping**

<b>Class</b>	<b>DiagnosticReadDTCInformation</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDTCInformation			
<b>Note</b>	This represents an instance of the "Read DTC Information" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticReadDtcInformations			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
readDTCInformationClass	DiagnosticReadDTCInformationClass	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 DiagnosticReadDTCInformation in the given context.

**Table A.93: DiagnosticReadDTCInformation**

<b>Class</b>	<b>DiagnosticReadDataByIdentifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
<b>Note</b>	This represents an instance of the "Read Data by Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticDataByIdentifiers			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticDataByIdentifier</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
readClass	<a href="#">DiagnosticReadDataByIdentifierClass</a>	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 DiagnosticReadDataByIdentifier in the given context.

**Table A.94: DiagnosticReadDataByIdentifier**

<b>Class</b>	<b>DiagnosticReadDataByIdentifierClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Read Data by Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticDataByIdentifiers			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maxDidToRead	PositiveInteger	0..1	attr	This attribute represents the maximum number of allowed DIDs in a single instance of DiagnosticReadDataByIdentifier.

**Table A.95: DiagnosticReadDataByIdentifierClass**

<b>Class</b>	<b>DiagnosticReadDataByPeriodicIDClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Read Data by periodic Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticReadDataByPeriodicIds			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
maxPeriodicDidToRead	PositiveInteger	0..1	attr	This represents the maximum number of data identifiers that can be included in one request.
periodicRate	<a href="#">DiagnosticPeriodicRate</a>	*	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.96: DiagnosticReadDataByPeriodicIDClass**

<b>Enumeration</b>	<b>DiagnosticRecordTriggerEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticFreezeFrame
<b>Note</b>	Triggers to allocate an event memory entry.
<b>Aggregated by</b>	<a href="#">DiagnosticExtendedDataRecord.trigger</a> , <a href="#">DiagnosticFreezeFrame.trigger</a>
<b>Literal</b>	<b>Description</b>
confirmed	capture on "Confirmed" <b>Tags:</b> atp.EnumerationLiteralIndex=0
custom	implement custom capture <b>Tags:</b> atp.EnumerationLiteralIndex=4
fdcThreshold	capture on "FDC Threshold" <b>Tags:</b> atp.EnumerationLiteralIndex=1
pending	capture on "Pending" <b>Tags:</b> atp.EnumerationLiteralIndex=2
testFailed	capture on "Test Failed" <b>Tags:</b> atp.EnumerationLiteralIndex=3

**Table A.97: DiagnosticRecordTriggerEnum**

<b>Class</b>	<b>DiagnosticRequestRoutineResults</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class represents the ability to define the result of a diagnostic routine execution.			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">DiagnosticRoutineSubfunction</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticRoutine.requestResult</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
request	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the request parameters.
response	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the response parameters.

**Table A.98: DiagnosticRequestRoutineResults**

<b>Class</b>	<b>DiagnosticResponseOnEvent</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
<b>Note</b>	This represents an instance of the "Response on Event" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticResponseOnEvents			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eventWindow	<a href="#">DiagnosticEventWindow</a>	*	aggr	This represents the applicable DiagnosticEventWindows
responseOnEventAction	<a href="#">DiagnosticResponseOnEventActionEnum</a>	0..1	attr	Defines sub-functions of the service ResponseOnEvent.
responseOnEventClass	<a href="#">DiagnosticResponseOnEventClass</a>	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.99: DiagnosticResponseOnEvent**

<b>Enumeration</b>	<b>DiagnosticResponseOnEventActionEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent
<b>Note</b>	This meta-class has the ability to define sub-functions of the UDS service ResponseOnEvent.
<b>Aggregated by</b>	<a href="#">DiagnosticResponseOnEvent.responseOnEventAction</a>
<b>Literal</b>	<b>Description</b>
clear	Clears the configured events. <b>Tags:</b> atp.EnumerationLiteralIndex=2
onChangeOfDataIdentifier	Reports based on change of data identifier. <b>Tags:</b> atp.EnumerationLiteralIndex=6
onComparisonOfValues	Triggered if data condition is met (e.g. RPM over 5000 1/min). <b>Tags:</b> atp.EnumerationLiteralIndex=8
onDTCStatusChange	Reports based on change of DTC status. <b>Tags:</b> atp.EnumerationLiteralIndex=7
report	Reports the activated events. <b>Tags:</b> atp.EnumerationLiteralIndex=3
reportDTCRecordInformationOnDtcStatusChange	Reports the DTC record-related information based on a DTC status change. (Subfunction 0x09) <b>Tags:</b> atp.EnumerationLiteralIndex=5
reportMostRecentDtcOnStatusChange	Triggers the report of the most recent failed or confirmed DTC (Subfunction 0x08). <b>Tags:</b> atp.EnumerationLiteralIndex=4
start	Starts the response on event service. <b>Tags:</b> atp.EnumerationLiteralIndex=1
stop	Stops the response on event service. <b>Tags:</b> atp.EnumerationLiteralIndex=0

**Table A.100: DiagnosticResponseOnEventActionEnum**

<b>Class</b>	<b>DiagnosticResponseOnEventClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
<b>Note</b>	This represents the ability to define common properties for all instances of the "Response on Event" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticResponseOnEvents			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.
maxSupportedDIDLLength	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.101: DiagnosticResponseOnEventClass**

<b>Enumeration</b>	<b>DiagnosticResponseToEcuResetEnum</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset		
<b>Note</b>	This enumeration controls the point in time in which a response to the reception of an EcuReset service shall be generated.		
<b>Aggregated by</b>	<a href="#">DiagnosticEcuResetClass.respondToReset</a>		
<b>Literal</b>	<b>Description</b>		
respondAfterReset	Answer to EcuReset service should come after the reset. <b>Tags:</b> atp.EnumerationLiteralIndex=0		
respondBeforeReset	Answer to EcuReset service should come before the reset. <b>Tags:</b> atp.EnumerationLiteralIndex=1		

**Table A.102: DiagnosticResponseToEcuResetEnum**

<b>Class</b>	<b>DiagnosticRoutine</b>		
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics		
<b>Note</b>	This meta-class represents the ability to define a diagnostic routine. <b>Tags:</b> atp.recommendedPackage=DiagnosticRoutines		
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>		
<b>Aggregated by</b>	ARPackage.element		





Class		DiagnosticRoutine		
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 <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
requestResult	<a href="#">DiagnosticRequest</a> <a href="#">RoutineResults</a>	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	<a href="#">DiagnosticStartRoutine</a>	0..1	aggr	This represents the ability to start a routine
stop	<a href="#">DiagnosticStopRoutine</a>	0..1	aggr	This represents the ability to stop a running routine.

**Table A.103: DiagnosticRoutine**

Class		DiagnosticRoutineGenericInterface		
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticAbstractRoutineInterface</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table A.104: DiagnosticRoutineGenericInterface**

Class		DiagnosticRoutineInterface		
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticAbstractRoutineInterface</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	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.105: DiagnosticRoutineInterface**

<b>Class</b>	<b>DiagnosticRoutineSubfunction</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This meta-class acts as an abstract base class to routine subfunctions.			
<b>Base</b>	ARObject, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">DiagnosticRequestRoutineResults</a> , <a href="#">DiagnosticStartRoutine</a> , <a href="#">DiagnosticStopRoutine</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
access Permission	<a href="#">DiagnosticAccess Permission</a>	0..1	ref	This reference represents the access permission of the owning routine subfunction.

**Table A.106: DiagnosticRoutineSubfunction**

<b>Class</b>	<b>DiagnosticSecurityAccess</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
<b>Note</b>	This represents an instance of the "Security Access" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticSecurityAccesses			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
requestSeedId	PositiveInteger	0..1	attr	This would be 0x01, 0x03, 0x05, ... The sendKey id can be computed by adding 1 to the requestSeedId
securityAccess Class	<a href="#">DiagnosticSecurity AccessClass</a>	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.
securityDelay TimeOnBoot	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	<a href="#">DiagnosticSecurityLevel</a>	0..1	ref	This reference identifies the applicable security level for the security access. <b>Stereotypes:</b> atp.Splittable <b>Tags:</b> atp.Splitkey=securityLevel

**Table A.107: DiagnosticSecurityAccess**

<b>Class</b>	<b>DiagnosticSecurityAccessClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticSecurityAccesses			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>







Class	DiagnosticSecurityAccessClass			
sharedTimer	Boolean	0..1	attr	Switch between separate or single shared timer instance and timer value. <ul style="list-style-type: none"> <li>• true: use shared timer instance and timer value for all security access levels combined.</li> <li>• false: use separate timer instance and timer values for each security level.</li> </ul>

**Table A.108: DiagnosticSecurityAccessClass**

Class	DiagnosticSecurityLevel			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	This meta-class represents the ability to define a security level considered for diagnostic purposes. <b>Tags:</b> atp.recommendedPackage=DiagnosticSecurityLevels			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	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.
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.109: DiagnosticSecurityLevel**

Class	DiagnosticSecurityLevelInterface			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

**Table A.110: DiagnosticSecurityLevelInterface**

Class	DiagnosticServiceClass (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
<b>Note</b>	This meta-class provides the ability to define common properties that are shared among all instances of sub-classes of DiagnosticServiceInstance.			





<b>Class</b>	<i>DiagnosticServiceClass</i> (abstract)			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Subclasses</b>	DiagnosticAuthenticationClass, DiagnosticClearDiagnosticInformationClass, DiagnosticClearResetEmissionRelatedInfoClass, DiagnosticComControlClass, DiagnosticControlDTCSettingClass, DiagnosticCustomServiceClass, DiagnosticDataTransferClass, <a href="#">DiagnosticDynamicallyDefineDataIdentifierClass</a> , <a href="#">DiagnosticEcuResetClass</a> , DiagnosticIoControlClass, DiagnosticReadDTCInformationClass, <a href="#">DiagnosticReadDataByIdentifierClass</a> , <a href="#">DiagnosticReadDataByPeriodicIDClass</a> , DiagnosticReadMemoryByAddressClass, DiagnosticReadScalingDataByIdentifierClass, DiagnosticRequestControlOfOnBoardDeviceClass, DiagnosticRequestCurrentPowertrainDataClass, DiagnosticRequestDownloadClass, DiagnosticRequestEmissionRelatedDTCClass, DiagnosticRequestEmissionRelatedDTCPermanentStatusClass, DiagnosticRequestFileTransferClass, DiagnosticRequestOnBoardMonitoringTestResultsClass, DiagnosticRequestPowertrainFreezeFrameDataClass, DiagnosticRequestUploadClass, DiagnosticRequestVehicleInfoClass, <a href="#">DiagnosticResponseOnEventClass</a> , DiagnosticRoutineControlClass, <a href="#">DiagnosticSecurityAccessClass</a> , <a href="#">DiagnosticSessionControlClass</a> , DiagnosticTransferExitClass, DiagnosticWriteDataByIdentifierClass, DiagnosticWriteMemoryByAddressClass			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.111: DiagnosticServiceClass**

<b>Class</b>	<b>DiagnosticServiceGenericMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
<b>Note</b>	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticServiceMappings			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnosticServiceInstance	<a href="#">DiagnosticServiceInstance</a>	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port.
pPortPrototypeInExecutable	<a href="#">PPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the DiagnosticServiceGenericMapping on the AUTOSAR adaptive platform. <b>Stereotypes:</b> atp.UriDef <b>InstanceRef implemented by:</b> 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. <b>Stereotypes:</b> atp.Splitable <b>Tags:</b> atp.Splitkey=process

**Table A.112: DiagnosticServiceGenericMapping**

<b>Class</b>	<i>DiagnosticServiceInstance</i> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
<b>Note</b>	This represents a concrete instance of a diagnostic service.			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			





<b>Class</b>	<b>DiagnosticServiceInstance</b> (abstract)			
<b>Subclasses</b>	<a href="#">DiagnosticAuthentication</a> , DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelatedInfo, <a href="#">DiagnosticComControl</a> , <a href="#">DiagnosticControlDTCSetting</a> , <a href="#">DiagnosticCustomServiceInstance</a> , <a href="#">DiagnosticDataByIdentifier</a> , <a href="#">DiagnosticDynamicallyDefineDataIdentifier</a> , <a href="#">DiagnosticEcuReset</a> , DiagnosticIOControl, <a href="#">DiagnosticMemoryByAddress</a> , <a href="#">DiagnosticReadDTCInformation</a> , DiagnosticReadDataByPeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequestPowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, <a href="#">DiagnosticResponseOnEvent</a> , DiagnosticRoutineControl, <a href="#">DiagnosticSecurityAccess</a> , DiagnosticSessionControl			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
accessPermission	<a href="#">DiagnosticAccessPermission</a>	0..1	ref	This represents the collection of DiagnosticAccessPermissions that allow for the execution of the referencing DiagnosticServiceInstance..  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=accessPermission
serviceClass	<a href="#">DiagnosticServiceClass</a>	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.  <b>Stereotypes:</b> atpAbstract

**Table A.113: DiagnosticServiceInstance**

<b>Class</b>	<b>DiagnosticServiceValidationConfiguration</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	This meta-class has the ability to configure the order of manufacturer/supplier-checks.  <b>Tags:</b> atp.recommendedPackage=DiagnosticValueConfigurations			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	<a href="#">SoftwareClusterDiagnosticDeploymentProps.validationConfiguration</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
manufacturerValidationOrder (ordered)	DiagnosticServiceValidationMapping	*	ref	This reference defines the order in which validations created by manufacturer are executed.
sovdValidationOrder (ordered)	<a href="#">DiagnosticSovdServiceValidationPortMapping</a>	*	ref	This reference defines the order in which validations of SOVD requests are executed.  <b>Tags:</b> atp.Status=candidate
supplierValidationOrder (ordered)	DiagnosticServiceValidationMapping	*	ref	This reference defines the order in which validations created by supplier are executed.

**Table A.114: DiagnosticServiceValidationConfiguration**

<b>Class</b>	<b>DiagnosticServiceValidationInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.115: DiagnosticServiceValidationInterface**

<b>Class</b>	<b>DiagnosticSession</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
<b>Note</b>	This meta-class represents the ability to define a diagnostic session. <b>Tags:</b> atp.recommendedPackage=DiagnosticSessions			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.
p2StarServer Max	TimeValue	0..1	attr	This is the session value for P2*ServerMax in seconds (per Session Control). The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds.

**Table A.116: DiagnosticSession**

<b>Class</b>	<b>DiagnosticSessionControlClass</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SessionControl			
<b>Note</b>	This meta-class contains attributes shared by all instances of the "Session Control" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticSessionControls			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticServiceClass</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
s3Server Timeout	TimeValue	0..1	attr	Time for the server to keep a diagnostic session other than the default session active while not receiving any diagnostic request message.

**Table A.117: DiagnosticSessionControlClass**

<b>Enumeration</b>	<b>DiagnosticSignificanceEnum</b>
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
<b>Note</b>	Significance level of a diagnostic event.
<b>Aggregated by</b>	<a href="#">DiagnosticTroubleCodeProps.significance</a>
<b>Literal</b>	<b>Description</b>
fault	Failure, which affects the component/ECU itself. <b>Tags:</b> atp.EnumerationLiteralIndex=0
occurrence	Issue, which indicates additional information concerning insufficient system behavior. <b>Tags:</b> atp.EnumerationLiteralIndex=1

**Table A.118: DiagnosticSignificanceEnum**

<b>Class</b>	<b>DiagnosticSovdLog</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
<b>Note</b>	This meta-class represents a "Log" SOVD service instance. <b>Tags:</b> atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
<b>Base</b>	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticSovdServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.119: DiagnosticSovdLog**

<b>Class</b>	<b>DiagnosticSovdMethod</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
<b>Note</b>	A DiagnosticSovdMethod represents a re-usable complex operation (that consists of primitive operations) in the context of the communication of an SOVD server. <b>Tags:</b> atp.Status=candidate atp.recommendedPackage=DiagnosticSovdMethods			
<b>Base</b>	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
delete	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive. <b>Tags:</b> atp.Status=candidate
get	DiagnosticSovdMethod Primitive	*	aggr	This represents the "get" method primitive. <b>Tags:</b> atp.Status=candidate
post	DiagnosticSovdMethod Primitive	*	aggr	This represents the "post" method primitive. <b>Tags:</b> atp.Status=candidate
put	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive. <b>Tags:</b> atp.Status=candidate

**Table A.120: DiagnosticSovdMethod**

<b>Class</b>	<b>DiagnosticSovdServiceValidationPortMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
<b>Note</b>	This mapping class identifies the PortPrototype in the application software that handles the SOVD service validation. <b>Tags:</b> atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <a href="#">DiagnosticMapping</a> , <a href="#">DiagnosticSwMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
pPortPrototype InExecutable	<a href="#">PPortPrototype</a>	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdValidationPortMapping on the AUTOSAR adaptive platform. <b>Stereotypes:</b> atpUriDef <b>Tags:</b> atp.Status=candidate <b>InstanceRef implemented by:</b> 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. <b>Tags:</b> atp.Status=candidate

**Table A.121: DiagnosticSovdServiceValidationPortMapping**

<b>Class</b>	<b>DiagnosticStartRoutine</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This represents the ability to start a diagnostic routine.			
<b>Base</b>	ARObject, <a href="#">DiagnosticRoutineSubfunction</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticRoutine.start</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
request	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the request parameters.
response	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the response parameters.

**Table A.122: DiagnosticStartRoutine**

<b>Enumeration</b>	<b>DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
<b>Note</b>	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.			
<b>Aggregated by</b>	<a href="#">DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear</a>			
<b>Literal</b>	<b>Description</b>			
statusBitAgingAnd Displacement	The "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies. <b>Tags:</b> atp.EnumerationLiteralIndex=0			
statusBitNormal	Aging and displacement has no impact on the "TestFailedSinceLastClear" status bits. <b>Tags:</b> atp.EnumerationLiteralIndex=1			

**Table A.123: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum**

<b>Class</b>	<b>DiagnosticStopRoutine</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
<b>Note</b>	This represents the ability to stop a diagnostic routine.			
<b>Base</b>	<i>ARObject</i> , <i>DiagnosticRoutineSubfunction</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
<b>Aggregated by</b>	<a href="#">DiagnosticRoutine.stop</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
request	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the request parameters.
response	<a href="#">DiagnosticParameter</a>	*	aggr	This represents the response parameters.

**Table A.124: DiagnosticStopRoutine**

<b>Class</b>	<b>DiagnosticTroubleCodeGroup</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
<b>Note</b>	The diagnostic trouble code group defines the DTCs belonging together and thereby forming a group. <b>Tags:</b> atp.recommendedPackage=DiagnosticTroubleCodes			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
dtc	DiagnosticTroubleCode	*	ref	This represents the collection of DiagnosticTroubleCodes defined by this DiagnosticTroubleCodeGroup. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=dtc.diagnosticTroubleCode, dtc.variationPoint.shortLabel vh.latestBindingTime=postBuild
groupNumber	PositiveInteger	0..1	attr	This represents the base number of the DTC group. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.125: DiagnosticTroubleCodeGroup**

<b>Class</b>	<b>DiagnosticTroubleCodeProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
<b>Note</b>	This element defines common Dtc properties that can be reused by different non OBD-relevant DTCs. <b>Tags:</b> atp.recommendedPackage=DiagnosticTroubleCodePropss			
<b>Base</b>	<i>ARElement</i> , <i>ARObject</i> , <i>CollectableElement</i> , <i>DiagnosticCommonElement</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
aging	<a href="#">DiagnosticAging</a>	0..1	ref	Reference to an aging algorithm in case that an aging/unlearning of the event is allowed. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=aging
diagnostic Memory	<a href="#">DiagnosticMemory Destination</a>	0..1	ref	Reference to the applicable DiagnosticMemory Destination. <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=diagnosticMemory





Class	DiagnosticTroubleCodeProps			
extendedDataRecord	<a href="#">DiagnosticExtendedDataRecord</a>	*	ref	Defines the links to an extended data class sampler. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=extendedDataRecord.diagnosticExtendedDataRecord, extendedDataRecord.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
freezeFrame	<a href="#">DiagnosticFreezeFrame</a>	*	ref	Define the links to a freeze frame class sampler. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> 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	<a href="#">DiagnosticDataIdentifierSet</a>	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. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=legislatedFreezeFrameContentUdsObd.diagnosticDataIdentifierSet, legislatedFreezeFrameContentUdsObd.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
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.
priority	PositiveInteger	0..1	attr	Priority of the event, in view of full event buffer. A lower value means higher priority. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
significance	<a href="#">DiagnosticSignificanceEnum</a>	0..1	attr	Significance of the event, which indicates additional information concerning fault classification and resolution.
snapshotRecordContent	<a href="#">DiagnosticDataIdentifierSet</a>	0..1	ref	This represents the freeze frame layout as a set of DIDs. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> atp.Splitkey=snapshotRecordContent.diagnosticDataIdentifierSet, snapshotRecordContent.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

**Table A.126: DiagnosticTroubleCodeProps**



<b>Class</b>	<b>DiagnosticTroubleCodeUds</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
<b>Note</b>	This element is used to describe non OBD-relevant DTCs. <b>Tags:</b> atp.recommendedPackage=DiagnosticTroubleCodes			
<b>Base</b>	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticTroubleCode, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
dtcProps	<a href="#">DiagnosticTroubleCodeProps</a>	0..1	ref	Defined properties associated with the DemDTC.
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. <b>Stereotypes:</b> atpSplittable; atpVariation <b>Tags:</b> 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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
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. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
severity	DiagnosticUdsSeverityEnum	0..1	attr	DTC severity according to ISO 14229-1. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime
wwhObdDtcClass	DiagnosticWwhObdDtcClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC. <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime

**Table A.127: DiagnosticTroubleCodeUds**

<b>Class</b>	<b>DiagnosticTroubleCodeUdsToClearConditionGroupMapping</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
<b>Note</b>	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. <b>Tags:</b> atp.recommendedPackage=DiagnosticMappings			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticMapping</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clearConditionGroup	DiagnosticClearConditionGroup	0..1	ref	This reference identifies the applicable DiagnosticClearConditionGroup.
troubleCodeUds	<a href="#">DiagnosticTroubleCodeUds</a>	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping.

**Table A.128: DiagnosticTroubleCodeUdsToClearConditionGroupMapping**

<b>Enumeration</b>	<b>DiagnosticTypeOfFreezeFrameRecordNumerationEnum</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
<b>Note</b>	FreezeFrame record numeration type			
<b>Aggregated by</b>	<a href="#">DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration</a>			
<b>Literal</b>	<b>Description</b>			
calculated	Freeze frame records will be numbered consecutive starting by 1 in their chronological order. <b>Tags:</b> atp.EnumerationLiteralIndex=0			
configured	Freeze frame records will be numbered based on the given configuration in their chronological order. <b>Tags:</b> atp.EnumerationLiteralIndex=1			

**Table A.129: DiagnosticTypeOfFreezeFrameRecordNumerationEnum**

<b>Class</b>	<b>DiagnosticUploadInterface</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
<b>Note</b>	This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform. <b>Tags:</b> atp.recommendedPackage=DiagnosticPortInterfaces			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticPortInterface</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">PortInterface</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.130: DiagnosticUploadInterface**

<b>Class</b>	<b>DiagnosticWriteDataByIdentifier</b>			
<b>Package</b>	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
<b>Note</b>	This represents an instance of the "Write Data by Identifier" diagnostic service. <b>Tags:</b> atp.recommendedPackage=DiagnosticDataByIdentifiers			
<b>Base</b>	<a href="#">ARElement</a> , <a href="#">ARObject</a> , <a href="#">CollectableElement</a> , <a href="#">DiagnosticCommonElement</a> , <a href="#">DiagnosticDataByIdentifier</a> , <a href="#">DiagnosticServiceInstance</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			





<b>Class</b>	<b>DiagnosticWriteDataByIdentifier</b>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
writeClass	DiagnosticWriteDataByIdentifierClass	0..1	ref	<p>This reference substantiates that abstract reference in the role serviceClass for this specific concrete class.</p> <p>Thereby, the reference represents the ability to access shared attributes among all DiagnosticWriteDataByIdentifier in the given context.</p>

**Table A.131: DiagnosticWriteDataByIdentifier**

<b>Class</b>	<b>DolpNetworkConfiguration</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
<b>Note</b>	This element collects DoIP properties that are network interface specific.			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	DolpInstantiation.networkInterface			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
eidRetrieval	DolpEidRetrievalEnum	0..1	attr	This attribute defines how DoIP Entity Identification is retrieved.
isActivationLineDependent	Boolean	0..1	attr	<p>This attribute defines whether the network interface</p> <ul style="list-style-type: none"> <li>is started "on-demand" when an activation line is sensed or</li> <li>is always available.</li> </ul>
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.
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.





Class	DolpNetworkConfiguration			
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_Do IP_Announce_Interval of ISO 13400-2:2019.
vehicle Identification SyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

**Table A.132: DolpNetworkConfiguration**

<b>Class</b>	<b>Identifiable</b> (abstract)
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
<b>Note</b>	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.
<b>Base</b>	<i>ARObject</i> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>
<b>Subclasses</b>	<p>ARPackage, <i>AbstractDolpLogicAddressProps</i>, <i>AbstractEvent</i>, <i>AbstractImplementationDataTypeElement</i>, <i>AbstractSecurityEventFilter</i>, <i>AbstractSecurityIdsmInstanceFilter</i>, <i>AbstractServiceInstance</i>, <i>AbstractSignalBasedToSignalTriggeringMapping</i>, <i>AdaptiveSwcInternalBehavior</i>, <i>ApApplicationEndpoint</i>, <i>ApplicationEndpoint</i>, <i>ApplicationError</i>, <i>AppliedStandard</i>, <i>ArtifactChecksum</i>, <i>ArtifactLocator</i>, <i>AtpBlueprint</i>, <i>AtpBlueprintable</i>, <i>AtpClassifier</i>, <i>AtpFeature</i>, <i>AutosarOperationArgumentInstance</i>, <i>AutosarVariableInstance</i>, <i>BuildActionEntity</i>, <i>BuildActionEnvironment</i>, <i>Chapter</i>, <i>CheckpointTransition</i>, <i>ClassContentConditional</i>, <i>ClientIdDefinition</i>, <i>ClientServerOperation</i>, <i>Code</i>, <i>CollectableElement</i>, <i>ComManagementMapping</i>, <i>CommConnectorPort</i>, <i>CommunicationConnector</i>, <i>CommunicationController</i>, <i>Compiler</i>, <i>ConsistencyNeeds</i>, <i>ConsumedEventGroup</i>, <i>CouplingPort</i>, <i>CouplingPortStructuralElement</i>, <i>CryptoCertificate</i>, <i>CryptoKeySlot</i>, <i>CryptoProvider</i>, <i>CryptoServiceMapping</i>, <i>DataPrototypeGroup</i>, <i>DataTransformation</i>, <i>DdsCpDomain</i>, <i>DdsCpPartition</i>, <i>DdsCpQosProfile</i>, <i>DdsCpTopic</i>, <i>DdsDomainRange</i>, <i>DependencyOnArtifact</i>, <i>DiagEventDebounceAlgorithm</i>, <i>DiagnosticAuthTransmitCertificateEvaluation</i>, <a href="#">DiagnosticConnectedIndicator</a>, <a href="#">DiagnosticDataElement</a>, <a href="#">DiagnosticDebounceAlgorithmProps</a>, <i>DiagnosticFunctionInhibitSource</i>, <i>DiagnosticParameterElement</i>, <a href="#">DiagnosticRoutineSubfunction</a>, <i>DiagnosticSovdMethodPrimitive</i>, <i>DltApplication</i>, <i>DltArgument</i>, <i>DltMessage</i>, <i>DolpInterface</i>, <i>DolpLogicAddress</i>, <i>DolpRoutingActivation</i>, <i>E2EProfileConfiguration</i>, <i>End2EndEventProtectionProps</i>, <i>End2EndMethodProtectionProps</i>, <i>EndToEndProtection</i>, <i>EthernetWakeupSleepOnDatelineConfig</i>, <i>EventHandler</i>, <i>EventMapping</i>, <i>ExclusiveArea</i>, <i>ExecutableEntity</i>, <i>ExecutionTime</i>, <i>FMAAttributeDef</i>, <i>FMFeatureMapAssertion</i>, <i>FMFeatureMapCondition</i>, <i>FMFeatureMapElement</i>, <i>FMFeatureRelation</i>, <i>FMFeatureRestriction</i>, <i>FMFeatureSelection</i>, <i>FieldMapping</i>, <i>FireAndForgetMethodMapping</i>, <i>FlexrayArTpNode</i>, <i>FlexrayTpPduPool</i>, <i>FrameTriggering</i>, <i>GeneralParameter</i>, <i>GlobalSupervision</i>, <i>GlobalTimeGateway</i>, <i>GlobalTimeMaster</i>, <i>GlobalTimeSlave</i>, <i>HealthChannel</i>, <i>HeapUsage</i>, <i>HwAttributeDef</i>, <i>HwAttributeLiteralDef</i>, <i>HwPin</i>, <i>HwPinGroup</i>, <i>IEEE1722TpAcfBus</i>, <i>IEEE1722TpAcfBusPart</i>, <i>IPSecRule</i>, <i>IPv6ExtHeaderFilterList</i>, <i>ISignalToIPduMapping</i>, <i>ISignalTriggering</i>, <i>IdentCaption</i>, <i>ImpositionTime</i>, <i>InternalTriggeringPoint</i>, <i>Keyword</i>, <i>LifeCycleState</i>, <i>Linker</i>, <i>MacMulticastGroup</i>, <i>MacSecKayParticipant</i>, <i>McDataInstance</i>, <i>MemorySection</i>, <i>MemoryUsage</i>, <i>MethodMapping</i>, <i>ModeDeclaration</i>, <i>ModeDeclarationMapping</i>, <i>ModeSwitchPoint</i>, <i>NetworkEndpoint</i>, <i>NmCluster</i>, <i>NmNode</i>, <i>PackageableElement</i>, <i>ParameterAccess</i>, <i>PduActivationRoutingGroup</i>, <i>PduToFrameMapping</i>, <i>PduTriggering</i>, <i>PerInstanceMemory</i>, <i>PersistencyDeploymentElement</i>, <i>PersistencyInterfaceElement</i>, <i>PhmSupervision</i>, <i>PhysicalChannel</i>, <i>PortGroup</i>, <i>PortInterfaceMapping</i>, <i>PossibleErrorReaction</i>, <i>ProcessToMachineMapping</i>, <i>Processor</i>, <i>ProcessorCore</i>, <i>PskIdentityToKeySlotMapping</i>, <i>ResourceConsumption</i>, <i>ResourceGroup</i>, <i>RootSwClusterDesignComponentPrototype</i>, <i>RootSwComponentPrototype</i>, <i>RootSwCompositionPrototype</i>, <i>RptComponent</i>, <i>RptContainer</i>, <i>RptExecutableEntity</i>, <i>RptExecutableEntityEvent</i>, <i>RptExecutionContext</i>, <i>RptProfile</i>, <i>RptServicePoint</i>, <i>RunnableEntityGroup</i>, <i>SdgAttribute</i>, <i>SdgClass</i>, <i>SecOcJobMapping</i>, <i>SecOcJobRequirement</i>, <i>SecureCommunicationAuthenticationProps</i>, <i>SecureCommunicationDeployment</i>, <i>SecureCommunicationFreshnessProps</i>, <i>SecurityEventContextProps</i>, <i>ServiceEventDeployment</i>, <i>ServiceFieldDeployment</i>, <i>ServiceInterfaceElementSecureComConfig</i>, <i>ServiceMethodDeployment</i>, <i>ServiceNeeds</i>, <i>SignalServiceTranslationEventProps</i>, <i>SignalServiceTranslationProps</i>, <i>SocketAddress</i>, <i>SoftwarePackageStep</i>, <i>SomeipEventGroup</i>, <i>SomeipProvidedEventGroup</i>, <i>SomeipTpChannel</i>, <i>SpecElementReference</i>, <i>StackUsage</i>, <i>StateManagementActionItem</i>, <i>StateManagementActionList</i>, <i>StateManagementStateNotification</i>, <i>StateManagementStateRequest</i>, <i>StaticSocketConnection</i>, <i>StructuredReq</i>, <i>SupervisionCheckpoint</i>, <i>SupervisionMode</i>, <i>SupervisionModeCondition</i>, <i>SwGenericAxisParamType</i>, <i>SwServiceArg</i>, <i>SwcServiceDependency</i>, <i>SystemMapping</i>, <i>TimeBaseResource</i>, <i>TimingClock</i>, <i>TimingClockSyncAccuracy</i>, <i>TimingCondition</i>, <i>TimingConstraint</i>, <i>Timing</i></p>





<b>Class</b>	<b>Identifiable</b> (abstract)			
	Description, TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, <i>TracedFailure</i> , TransformationProps, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> 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.  <b>Tags:</b> 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.  <b>Tags:</b> 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".  <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp.  <b>Tags:</b> xml.attribute=true

**Table A.133: Identifiable**

<b>Class</b>	<b>ImplementationProps</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::CommonStructure::Implementation			
<b>Note</b>	Defines a symbol to be used as (depending on the concrete case) either a complete replacement or a prefix when generating code artifacts.			
<b>Base</b>	ARObject, <a href="#">Referrable</a>			
<b>Subclasses</b>	BswSchedulerNamePrefix, ExecutableEntityActivationReason, SectionNamePrefix, <a href="#">SymbolProps</a> , SymbolicNameProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
symbol	CIdentifier	0..1	attr	The symbol to be used as (depending on the concrete case) either a complete replacement or a prefix.

**Table A.134: ImplementationProps**

<b>Class</b>	<b>MultilanguageReferrable</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	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.			
<b>Base</b>	ARObject, <a href="#">Referrable</a>			
<b>Subclasses</b>	Caption, DefItem, DocumentationContext, <a href="#">Identifiable</a> , SdgCaption, <a href="#">TraceReferrable</a> , <a href="#">Traceable</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.135: MultilanguageReferrable**

<b>Class</b>	<b>PPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port providing a certain port interface.			
<b>Base</b>	ARObject, <a href="#">AbstractProvidedPortPrototype</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpFeature</a> , <a href="#">AtpPrototype</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	<a href="#">AtpClassifier.atpFeature</a> , <a href="#">SwComponentType.port</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
provided Interface	<a href="#">PortInterface</a>	0..1	tref	The interface that this port provides. <b>Stereotypes:</b> isOfType

**Table A.136: PPortPrototype**

<b>Class</b>	<b>PortInterface</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
<b>Note</b>	Abstract base class for an interface that is either provided or required by a port of a software component.			
<b>Base</b>	ARElement, ARObject, <a href="#">AtpBlueprint</a> , <a href="#">AtpBlueprintable</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpType</a> , <a href="#">CollectableElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PackageableElement</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">AbstractRawDataStreamInterface</a> , <a href="#">AbstractSynchronizedTimeBaseInterface</a> , ClientServerInterface, <a href="#">CryptoInterface</a> , <a href="#">DataInterface</a> , <a href="#">DiagnosticPortInterface</a> , FirewallStateSwitchInterface, <a href="#">IdsmAbstractPortInterface</a> , LogAndTraceInterface, ModeSwitchInterface, NetworkManagementPortInterface, <a href="#">PersistenceInterface</a> , <a href="#">PlatformHealthManagementInterface</a> , ServiceInterface, <a href="#">StateManagementPortInterface</a> , TriggerInterface			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





<b>Class</b>	<b>PortInterface</b> (abstract)			
namespace (ordered)	<a href="#">SymbolProps</a>	*	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.  <b>Stereotypes:</b> atpSplittable <b>Tags:</b> atp.Splitkey=namespace.shortName

**Table A.137: PortInterface**

<b>Class</b>	<b>PortPrototype</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	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.			
<b>Base</b>	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	AbstractProvidedPortPrototype, AbstractRequiredPortPrototype			
<b>Aggregated by</b>	AtpClassifier.atpFeature, SwComponentType.port			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServer Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non volatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

**Table A.138: PortPrototype**

<b>Class</b>	<b>RPortPrototype</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	Component port requiring a certain port interface.			
<b>Base</b>	ARObject, AbstractRequiredPortPrototype, AtpBlueprintable, AtpFeature, AtpPrototype, <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">PortPrototype</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	AtpClassifier.atpFeature, SwComponentType.port			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
required Interface	<a href="#">PortInterface</a>	0..1	tref	The interface that this port requires.  <b>Stereotypes:</b> isOfType

**Table A.139: RPortPrototype**

<b>Class</b>	<b>Referrable</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
<b>Note</b>	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
<b>Base</b>	ARObject			
<b>Subclasses</b>	<i>AtpDefinition</i> , <i>BswDistinguishedPartition</i> , <i>BswModuleCallPoint</i> , <i>BswModuleClientServerEntry</i> , <i>BswVariableAccess</i> , <i>CouplingPortTrafficClassAssignment</i> , <i>CppImplementationDataTypeContextTarget</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>SocketConnectionBundle</i> , <i>SomeipRequiredEventGroup</i> , <i>TimeSyncServerConfiguration</i> , <i>TpConnectionIdent</i>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpIdentityContributor <b>Tags:</b> xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments.  <b>Tags:</b> xml.sequenceOffset=-90

**Table A.140: Referrable**

<b>Class</b>	<b>SoftwareCluster</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	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.  <b>Tags:</b> atp.recommendedPackage=SoftwareClusters			
<b>Base</b>	<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>			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
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.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=artifactChecksum.shortName, artifactChecksum.uri
artifactLocator	ArtifactLocator	*	aggr	This aggregation represents the artifact locations that are relevant in the context of the enclosing SoftwareCluster
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=conflictsTo







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.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=containedARElement
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster.
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=containedPackageElement
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster.
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=dependsOn
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all SoftwareClusterDesigns applicable for the enclosing SoftwareCluster.  <b>Stereotypes:</b> atpUriDef
diagnostic Deployment Props	<a href="#">SoftwareCluster DiagnosticDeployment Props</a>	0..1	ref	This reference identifies the applicable SoftwareClusterDiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster.
installation Behavior	SoftwareCluster InstallationBehavior Enum	0..1	attr	This attribute controls the behavior of the SoftwareCluster in terms of installation.
license	Documentation	*	ref	This attribute allows for the inclusion of the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license.
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster.  <b>Stereotypes:</b> atpSplitable <b>Tags:</b> atp.Splitkey=moduleInstantiation
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test.
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature.
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

**Table A.141: SoftwareCluster**

<b>Class</b>	<b>SoftwareClusterDiagnosticAddress</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	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. ...).			
<b>Base</b>	ARObject			
<b>Subclasses</b>	SoftwareClusterSovdAddress, SoftwareClusterUdsDiagnosticAddress			
<b>Aggregated by</b>	DiagnosticCommonProps.diagnosticAddress			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
address Semantics	SoftwareClusterDiagnosticAddress SemanticsEnum	0..1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

**Table A.142: SoftwareClusterDiagnosticAddress**

<b>Enumeration</b>	<b>SoftwareClusterDiagnosticAddressSemanticsEnum</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster.			
<b>Aggregated by</b>	SoftwareClusterDiagnosticAddress.addressSemantics			
<b>Literal</b>	<b>Description</b>			
functionalAddress	This address represents a functional address. <b>Tags:</b> atp.EnumerationLiteralIndex=1			
physicalAddress	This address represents a physical address. <b>Tags:</b> atp.EnumerationLiteralIndex=0			

**Table A.143: SoftwareClusterDiagnosticAddressSemanticsEnum**

<b>Class</b>	<b>SoftwareClusterDiagnosticDeploymentProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	This meta-class acts as the owner of all deployment-related diagnostic properties of a SoftwareCluster. <b>Tags:</b> atp.recommendedPackage=SoftwareClusterDiagnosticProps			
<b>Base</b>	ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeploymentElement, UploadablePackageElement			
<b>Aggregated by</b>	ARPackage.element			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
diagnostic Extract	DiagnosticContributionSet	0..1	ref	This reference identifies the DiagnosticContributionSet that is applicable for the referencing SoftwareCluster.
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.144: SoftwareClusterDiagnosticDeploymentProps**

<b>Class</b>	<b>SoftwareClusterSovdAddress</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
<b>Note</b>	This meta-class represents the ability to define a diagnostic address specifically for the SOVD case. <b>Tags:</b> atp.Status=candidate			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">SoftwareClusterDiagnosticAddress</a>			
<b>Aggregated by</b>	<a href="#">DiagnosticCommonProps.diagnosticAddress</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
component Qualifier	String	0..1	attr	This attribute is used to specify the component qualifier for the usage in an SOVD query. <b>Tags:</b> atp.Status=candidate

**Table A.145: SoftwareClusterSovdAddress**

<b>Class</b>	<b>SovdModuleInstantiation</b> (abstract)			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
<b>Note</b>	This abstract meta-class serves as the base class for meta-classes that describe the configuration of an SOVD module. <b>Tags:</b> atp.Status=candidate			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AdaptiveModuleInstantiation</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">NonOsModuleInstantiation</a> , <a href="#">Referrable</a>			
<b>Subclasses</b>	<a href="#">SovdGatewayInstantiation</a> , <a href="#">SovdServerInstantiation</a>			
<b>Aggregated by</b>	<a href="#">AtpClassifier.atpFeature</a> , <a href="#">Machine.moduleInstantiation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
communication Connector	Communication Connector	*	ref	This reference identifies the collection of communication connectors used by the SOVD module instantiation for vehicle-internal communication. <b>Tags:</b> 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. <b>Tags:</b> 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. <b>Tags:</b> 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. <b>Tags:</b> atp.Status=candidate

**Table A.146: SovdModuleInstantiation**

<b>Class</b>	<b>SovdServerInstantiation</b>			
<b>Package</b>	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
<b>Note</b>	This meta-class represents the configuration of an SOVD server. <b>Tags:</b> atp.Status=candidate			
<b>Base</b>	<a href="#">ARObject</a> , <a href="#">AdaptiveModuleInstantiation</a> , <a href="#">AtpClassifier</a> , <a href="#">AtpFeature</a> , <a href="#">AtpStructureElement</a> , <a href="#">Identifiable</a> , <a href="#">MultilanguageReferrable</a> , <a href="#">NonOsModuleInstantiation</a> , <a href="#">Referrable</a> , <a href="#">SovdModuleInstantiation</a>			
<b>Aggregated by</b>	<a href="#">AtpClassifier.atpFeature</a> , <a href="#">Machine.moduleInstantiation</a>			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>





<b>Class</b>	<b>SovdServerInstantiation</b>			
component Qualifier	String	0..1	attr	This attributes described the component qualifier used to compose an SOVD query. <b>Tags:</b> atp.Status=candidate

**Table A.147: SovdServerInstantiation**

<b>Class</b>	<<atpVariation>> <b>SwDataDefProps</b>			
<b>Package</b>	M2::MSR::DataDictionary::DataDefProperties			
<b>Note</b>	<p>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.</p> <p>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.</p> <p>SwDataDefProps covers various aspects:</p> <ul style="list-style-type: none"> <li>• 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</li> <li>• Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddr Method, swPointerTargetProps, baseType, implementationDataType and additionalNativeTypeQualifier</li> <li>• Access policy for the MCD system, mainly expressed by swCalibrationAccess</li> <li>• Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalid Value</li> <li>• Code generation policy provided by swRecordLayout</li> </ul> <p><b>Tags:</b> vh.latestBindingTime=codeGenerationTime</p>			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	AutosarDataType.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, DataPrototype.swDataDefProps, DataPrototypeTransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, DiagnosticEnvDataElementCondition.swDataDefProps, DltArgument.networkRepresentation, FlatInstanceDescriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, ReceiverComSpec.networkRepresentation, SenderComSpec.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, SystemSignal.physicalProps			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
additionalNativeTypeQualifier	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. <b>Tags:</b> xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. <b>Tags:</b> xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false





<b>Class</b>	<b>&lt;&lt;atpVariation&gt;&gt; SwDataDefProps</b>			
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. <b>Tags:</b> xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. <b>Tags:</b> xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially <ul style="list-style-type: none"> <li>• redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype</li> <li>• the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly</li> <li>• the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly</li> <li>• the data type of an SwServiceArg, if it does not refer to a base type directly</li> </ul> <b>Tags:</b> xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. <b>Tags:</b> 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. <b>Tags:</b> 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. <b>Tags:</b> xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. <b>Tags:</b> xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. <b>Tags:</b> xml.sequenceOffset=70





<b>Class</b>	<b>&lt;&lt;atpVariation&gt;&gt; SwDataDefProps</b>			
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. <b>Tags:</b> xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. <b>Tags:</b> 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). <b>Tags:</b> xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. <b>Tags:</b> xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. <b>Tags:</b> xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process.  The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula).  In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution.  The resolution is specified in the physical domain according to the property "unit". <b>Tags:</b> xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. <b>Tags:</b> 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 . <b>Tags:</b> xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object.  Note: This atpSplittable property has no atp.Splitkey due to atpVariation (PropertySetPattern). <b>Stereotypes:</b> atpSplittable <b>Tags:</b> xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. <b>Tags:</b> xml.sequenceOffset=290





<b>Class</b>	<b>&lt;&lt;atpVariation&gt;&gt; SwDataDefProps</b>			
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system.  So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing.  <b>Tags:</b> xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object.  <b>Tags:</b> xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block  <b>Stereotypes:</b> atpVariation <b>Tags:</b> vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension.  The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on.  For one-dimensional value blocks the attribute swValueBlockSize shall be used and this attribute shall not exist.  <b>Stereotypes:</b> atpVariation <b>Tags:</b> 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.  <b>Tags:</b> xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType.  <b>Tags:</b> xml.sequenceOffset=355

**Table A.148: SwDataDefProps**

<b>Class</b>	<b>SymbolProps</b>			
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Components			
<b>Note</b>	This meta-class represents the ability to contribute a part of a namespace.			
<b>Base</b>	ARObject, <a href="#">ImplementationProps</a> , <a href="#">Referrable</a>			
<b>Aggregated by</b>	Allocator.namespace, ApApplicationErrorDomain.namespace, <a href="#">AtomicSwComponentType.symbolProps</a> , <a href="#">CppImplementationDataType.namespace</a> , ImplementationDataType.symbolProps, <a href="#">PortInterface.namespace</a> , SecurityEventDefinition.eventSymbolName			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
–	–	–	–	–

**Table A.149: SymbolProps**

<b>Class</b>	<b>ValueGroup</b>			
<b>Package</b>	M2::MSR::CalibrationData::CalibrationValue			
<b>Note</b>	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.			
<b>Base</b>	ARObject			
<b>Aggregated by</b>	SwValues.vg			
<b>Attribute</b>	<b>Type</b>	<b>Mult.</b>	<b>Kind</b>	<b>Note</b>
label	MultilanguageLong Name	0..1	aggr	This label allows to give the valueGroup a particular name. It can be used if the Values are rendered as a table. <b>Tags:</b> xml.sequenceOffset=20
vgContents	SwValues	0..1	aggr	This represents the contents of the value group. <b>Tags:</b> xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

**Table A.150: ValueGroup**



## B Platform Extension API (normative)

The focus of the APIs in this section are for OEM-specific platform extensions. The abstraction of the interfaces is lower which could lead to a higher machine dependency.

### B.1 C++ UDS Transportlayer API Interfaces

This chapter lists all provided and required C++ API interfaces of the [DM](#) for interaction with a UDS Transportlayer implementation.

#### B.1.1 UDS Transportlayer Types

##### B.1.1.1 uds\_transport::ByteSpan

**[SWS\_DM\_00338]{DRAFT} Definition of API type ara::diag::uds\_transport::ByteSpan** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_types.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	ByteSpan
<b>Syntax:</b>	using ByteSpan = ara::core::Span<ara::core::Byte>;
<b>Description:</b>	This is the type of ByteSpan.

]([RS\\_Diag\\_04147](#))

##### B.1.1.2 uds\_transport::ChannelID

**[SWS\_DM\_00337]{DRAFT} Definition of API type ara::diag::uds\_transport::ChannelID** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_types.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	ChannelID
<b>Syntax:</b>	using ChannelID = uint32_t;
<b>Description:</b>	This is the identification of channel on which to transmit.

]([RS\\_Diag\\_04147](#))

### B.1.1.3 uds\_transport::Priority

[SWS\_DM\_00451]{DRAFT} **Definition of API type ara::diag::uds\_transport::Priority** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_types.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	Priority
<b>Syntax:</b>	using Priority = uint8_t;
<b>Description:</b>	This is the Priority of the given message, used for prioritization of conversations.

]([RS\\_Diag\\_04147](#))

### B.1.1.4 uds\_transport::ProtocolKind

[SWS\_DM\_00452]{DRAFT} **Definition of API type ara::diag::uds\_transport::ProtocolKind** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_types.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	ProtocolKind
<b>Syntax:</b>	using ProtocolKind = ara::core::String;
<b>Description:</b>	This is the identifier of protocol kind associated to message

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.1.5 uds\_transport::UdsMessageConstPtr

[SWS\_DM\_00304]{DRAFT} **Definition of API type ara::diag::uds\_transport::UdsMessageConstPtr** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsMessageConstPtr
<b>Syntax:</b>	using UdsMessageConstPtr = std::unique_ptr<const UdsMessage, std::function<void(const UdsMessage*)> >;
<b>Description:</b>	This is the unique_ptr for constant UdsMessages containing a custom deleter as provided by the generic/core DM part towards the UdsTransportLayer-Plugin.
<b>Notes:</b>	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: UdsMessageConstPtr shall behave like a std::unique_ptr<const UdsMessage>!

]([RS\\_Diag\\_04147](#))

### B.1.1.6 uds\_transport::UdsMessagePtr

[SWS\_DM\_00303]{DRAFT} Definition of API type ara::diag::uds\_transport::UdsMessagePtr [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsMessagePtr
<b>Syntax:</b>	using UdsMessagePtr = std::unique_ptr<UdsMessage, std::function<void(UdsMessage*)> >;
<b>Description:</b>	This is the unique_ptr for UdsMessages containing a custom deleter as provided by the generic/core DM part towards the UdsTransportLayer-Plugin.
<b>Notes:</b>	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>!

]([RS\\_Diag\\_04147](#))

### B.1.1.7 uds\_transport::UdsTransportProtocolHandlerID

[SWS\_DM\_00336]{DRAFT} Definition of API type ara::diag::uds\_transport::UdsTransportProtocolHandlerID [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_types.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsTransportProtocolHandlerID
<b>Syntax:</b>	using UdsTransportProtocolHandlerID = uint8_t;
<b>Description:</b>	UdsTransportProtocolHandler are flexible "plugins", which need an identification.

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

## B.1.2 UdsMessage Class

[SWS\_DM\_00291]{DRAFT} Definition of API class ara::diag::uds\_transport::UdsMessage [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsMessage
<b>Syntax:</b>	class UdsMessage {...};





<b>Description:</b>	class represents an UDS message exchanged between DM generic core (UdsTransportProtocol Mgr) and a specific implementation of UdsTransportProtocolHandler on diagnostic request reception path or diagnostic response transmission path.  UdsMessage provides the storage for UDS requests/responses. Instances of UdsMessage (with optimized resource allocation) are only created by DM generic core. UdsTransportProtocol Handler read/write on it.
---------------------	--

]([RS\\_Diag\\_04147](#))

## B.1.2.1 UdsMessage Types

### B.1.2.1.1 uds\_transport::UdsMessage::Address

**[SWS\_DM\_00293]{DRAFT} Definition of API type ara::diag::uds\_transport::UdsMessage::Address** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsMessage</a>
<b>Symbol:</b>	Address
<b>Syntax:</b>	using Address = uint16_t;
<b>Description:</b>	type for UDS source and target addresses

]([RS\\_Diag\\_04147](#))

### B.1.2.1.2 uds\_transport::UdsMessage::MetaInfoMap

**[SWS\_DM\_00294]{DRAFT} Definition of API type ara::diag::uds\_transport::UdsMessage::MetaInfoMap** [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsMessage</a>
<b>Symbol:</b>	MetaInfoMap
<b>Syntax:</b>	using MetaInfoMap = ara::core::Map<ara::core::String, ara::core::String>;
<b>Description:</b>	Type for the meta information attached to a UdsMessage. .

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04170](#))

### B.1.2.1.3 uds\_transport::UdsMessage::TargetAddressType

[SWS\_DM\_00296]{DRAFT} Definition of API enum ara::diag::uds\_transport::UdsMessage::TargetAddressType [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	TargetAddressType	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class TargetAddressType : std::uint8_t {...};	
<b>Values:</b>	kPhysical= 0	–
	kFunctional= 1	–
<b>Description:</b>	type of target address in UdsMessage	

]([RS\\_Diag\\_04147](#))

### B.1.2.2 UdsMessage Methods

#### B.1.2.2.1 uds\_transport::UdsMessage::UdsMessage constructor

[SWS\_DM\_09012]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::UdsMessage [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	UdsMessage()	
<b>Syntax:</b>	UdsMessage ();	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	non public default ctor. The default ctor is protected as we want to forbid, that UdsTransport Protocol handlers do create UdsMessages on its own! Only DM is allowed to create and hands over UdsMessagePtrs to UdsTransportProtocolHandler.	
<b>Visibility:</b>	protected	

]([RS\\_Diag\\_04147](#))

### B.1.2.2.2 uds\_transport::UdsMessage::~~UdsMessage

[SWS\_DM\_09010]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::~~UdsMessage [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsMessage</a>
<b>Symbol:</b>	~UdsMessage()
<b>Syntax:</b>	virtual ~UdsMessage ()=default;
<b>Description:</b>	Destructing the uds message.

]([RS\\_Diag\\_04147](#))

### B.1.2.2.3 uds\_transport::UdsMessage::AddMetaInfo

[SWS\_DM\_00302]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::AddMetaInfo [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsMessage</a>
<b>Symbol:</b>	AddMetaInfo(MetaInfoMap metaInfo)
<b>Syntax:</b>	virtual void AddMetaInfo (MetaInfoMap metaInfo);
<b>Parameters (in):</b>	metaInfo      meta information relevant for UdsMessage
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Thread Safety:</b>	unsafe
<b>Description:</b>	add new metaInfo to this message.
<b>Notes:</b>	typically called by the transport plugin to add channel specific meta-info. (see SWS - there are already predefined meta-info keys for DoIP...)

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04170](#))

### B.1.2.2.4 uds\_transport::UdsMessage::GetPayload

[SWS\_DM\_00300]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::GetPayload [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsMessage</a>
<b>Symbol:</b>	GetPayload()
<b>Syntax:</b>	virtual const ara::core::Span< const uint8_t > & GetPayload () const;





<b>Return value:</b>	const ara::core::Span< const uint8_t > &	The entire payload (A_Data) with no write access.
<b>Exception Safety:</b>	not exception safe	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	Get the UDS message data starting with the SID (A_Data as per ISO) with no write access.	
<b>Notes:</b>	marked as "unsafe" with regard to threadsafety as implementation is allowed to do ressource allocation of buffer in the context of this call.	

](RS\_Diag\_04147)

### [SWS\_DM\_00301]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::GetPayload [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	GetPayload()	
<b>Syntax:</b>	virtual uds_transport::ByteSpan & GetPayload ();	
<b>Return value:</b>	uds_transport::ByteSpan &	payload of the UdsMessage starting from SID.
<b>Exception Safety:</b>	not exception safe	
<b>Thread Safety:</b>	unsafe	
<b>Description:</b>	return the underlying buffer for write access.	
<b>Notes:</b>	needed by UdsTransportProtocolHandler impl. to fill the UdsMessage with data in RX path. I.e. UdsTransportProtocolHandler impl. gets the UdsMessage instance from call to UdsTransport ProtocolMgr::IndicateMessage() and then calls this method on it and write into returned uds_transport::ByteVector.  marked as "unsafe" with regard to threadsafety as implementation is allowed to do ressource allocation of buffer in the context of this call.	

](RS\_Diag\_04147)

### B.1.2.2.5 uds\_transport::UdsMessage::GetSa

### [SWS\_DM\_00297]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::GetSa [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	GetSa()	
<b>Syntax:</b>	virtual Address GetSa () const noexcept;	
<b>Return value:</b>	Address	The source address of the uds message.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	reentrant	
<b>Description:</b>	Get the source address of the uds message.	

](RS\_Diag\_04147, RS\_Diag\_04174)

### B.1.2.2.6 uds\_transport::UdsMessage::GetTa

[SWS\_DM\_00298]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::GetTa [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	GetTa()	
<b>Syntax:</b>	virtual Address GetTa () const noexcept;	
<b>Return value:</b>	Address	The target address of the uds message.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	reentrant	
<b>Description:</b>	Get the target address of the uds message.	

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04174](#))

### B.1.2.2.7 uds\_transport::UdsMessage::GetTaType

[SWS\_DM\_00299]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsMessage::GetTaType [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/uds_message.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsMessage	
<b>Symbol:</b>	GetTaType()	
<b>Syntax:</b>	virtual TargetAddressType GetTaType () const noexcept;	
<b>Return value:</b>	TargetAddressType	The target address type of the uds message.
<b>Exception Safety:</b>	noexcept	
<b>Thread Safety:</b>	reentrant	
<b>Description:</b>	Get the target address type (phys/func) of the uds message.	

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04174](#))

## B.1.3 UdsTransportProtocolHandler Class

[SWS\_DM\_00315]{DRAFT} Definition of API class ara::diag::uds\_transport::UdsTransportProtocolHandler [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	namespace ara::diag::uds_transport	
<b>Symbol:</b>	UdsTransportProtocolHandler	







<b>Syntax:</b>	<code>class UdsTransportProtocolHandler {...};</code>
<b>Description:</b>	Abstract Class, which a specific UDS Transport Protocol (plugin) shall subclass.

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.3.1 UdsTransportProtocolHandlerClass Types

#### B.1.3.1.1 uds\_transport::UdsTransportProtocolHandler::InitializationResult

[SWS\_DM\_09017]{DRAFT} Definition of API enum `ara::diag::uds_transport::UdsTransportProtocolHandler::InitializationResult` [

<b>Kind:</b>	enumeration				
<b>Header file:</b>	<code>#include "ara/diag/uds_transport/protocol_handler.h"</code>				
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>				
<b>Scope:</b>	<code>class ara::diag::uds_transport::UdsTransportProtocolHandler</code>				
<b>Symbol:</b>	InitializationResult				
<b>Underlying type:</b>	<code>std::uint8_t</code>				
<b>Syntax:</b>	<code>enum class InitializationResult : std::uint8_t {...};</code>				
<b>Values:</b>	<table border="1"> <tr> <td><code>kInitializeOk= 0</code></td> <td>–</td> </tr> <tr> <td><code>kInitializeFailed= 1</code></td> <td>–</td> </tr> </table>	<code>kInitializeOk= 0</code>	–	<code>kInitializeFailed= 1</code>	–
<code>kInitializeOk= 0</code>	–				
<code>kInitializeFailed= 1</code>	–				
<b>Description:</b>	Result of Initialize handler.				

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.3.2 UdsTransportProtocolHandler Methods

#### B.1.3.2.1 uds\_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler

[SWS\_DM\_09015]{DRAFT} Definition of API function `ara::diag::uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/uds_transport/protocol_handler.h"</code>
<b>Scope:</b>	<code>class ara::diag::uds_transport::UdsTransportProtocolHandler</code>
<b>Symbol:</b>	<code>UdsTransportProtocolHandler(const UdsTransportProtocolHandlerID handlerId, UdsTransportProtocolMgr &amp;transportProtocolMgr)</code>
<b>Syntax:</b>	<code>explicit UdsTransportProtocolHandler (const <a href="#">UdsTransportProtocolHandlerID</a> handlerId, <a href="#">UdsTransportProtocolMgr</a> &amp;transportProtocolMgr);</code>





<b>Parameters (in):</b>	handlerId	the handler ID used by DM to identify this handler. This is just a number/identification given by the DM core when instantiating a UdsTransportProtocolHandler instance to be able to distinguish it from other handler-plugins or built-in UdsTransportProtocolHandler implementations.
	transportProtocolMgr	reference to UdsTransportProtocolMgr owned by this DM, with which UdsTransportProtocolHandler instance shall interact.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of UdsTransportProtocolHandler.	

|(RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.3.2.2 uds\_transport::UdsTransportProtocolHandler::~~UdsTransport

[SWS\_DM\_09016]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::~~UdsTransportProtocolHandler [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	~UdsTransportProtocolHandler()	
<b>Syntax:</b>	virtual ~UdsTransportProtocolHandler ()=default;	
<b>Description:</b>	Destructor of UdsTransportProtocolHandler.	

|(RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.3.2.3 uds\_transport::UdsTransportProtocolHandler::GetHandlerID

[SWS\_DM\_00325]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::GetHandlerID [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	GetHandlerID()	
<b>Syntax:</b>	virtual UdsTransportProtocolHandlerID GetHandlerID () const;	
<b>Return value:</b>	UdsTransportProtocolHandlerID	UdsTransportProtocolHandlerID.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Return the UdsTransportProtocolHandlerID, which was given to the implementation during construction (ctor call).	

|(RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.3.2.4 uds\_transport::UdsTransportProtocolHandler::Initialize

[SWS\_DM\_00319]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::Initialize [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	Initialize()	
<b>Syntax:</b>	virtual InitializationResult Initialize ()=0;	
<b>Return value:</b>	InitializationResult	kInitializeOk if initialization was successful, else kInitializeFailed.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Initializes UDS transport protocol handler. This methods needs to be called before a call of Start(). It is meant to separate the actual initialization of the transport protocol handler from the constructor.	

] ([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.3.2.5 uds\_transport::UdsTransportProtocolHandler::NotifyReestablishment

[SWS\_DM\_00326]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::NotifyReestablishment [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	NotifyReestablishment(ChannelID channelId)	
<b>Syntax:</b>	virtual bool NotifyReestablishment (ChannelID channelId)=0;	
<b>Parameters (in):</b>	channelId	channelID, whose re-establishment shall be notified to Uds TransportProtocolMgr
<b>Return value:</b>	bool	true if notification request is accepted and can be fulfilled.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Tells the UdsTransportProtocolHandler, to notify the DM core via UdsTransportProtocol Mgr::ChannelReestablished if the given channel has been re-established after next Uds TransportProtocolHandler::Start. The main purpose of this method is to allow the DM to provide an ECU-Reset (0x11 service), with configuration option "positive response AFTER reset". In this scenario the request for 0x11 will be received on a certain channel with identifying tuple <p_x, c_y> (GlobalChannelIdentifier). Then the ECU-Reset takes place and after ECU-Restart all Uds ProtocolHandlers/plugins get restarted via call to UdsTransportProtocolHandler::Start(). Now there are two expectations, when this method has been called before and returned "true": IF the same remote client connects to the UdsProtocolHandler, it shall get a channel identification with the same identifying tuple <p_x, c_y> as the last time. It consecutively calls UdsTransport ProtocolMgr::ChannelReestablished(GlobalChannelIdentifier<p_x, c_y>).	





<b>Notes:</b>	: The UdsTransportProtocolHandler will call UdsTransportProtocolMgr::ChannelReestablished() after UdsTransportProtocolHandler::Start() is called and as soon as it detects that the remote client/tester is reachable again. The detection/decision, whether the "same" client reconnects as before is an UdsProtocol Handler implementation specific decision. The general expectation is: If the channel is set up from exactly the same remote network-endpoint, it typically gets the same channelID (c_y part of the tuple). 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 channel ID (c_y part of the tuple) can be reassigned. This is the case if the mapping of protocol specific channel info -> channelID isn't a stable bijective mapping! Small example: The underlying network protocol, which UdsProtocolHandler implements is based on TCP. At the point in time, where the 0x11 SI request is received on channel identified by <p_x, c_y> the DM calls Notify Reestablishment() on this channelID. Now the implementation of UdsProtocolHandler stores non- volatile in the context of this call: the NetworkEndpoint (IP-address and port number) of the channel the NetworkEndpoint (IP-address and port number) of the local port (because in this example, the UdsTransportProtocolHandler listens on/supports different ports) the channelID (c_y part) it has currently assigned. After restart this channelID is only reused for a channel with exactly the same NetworkEndpoint addresses as stored non-volatile. If this channelID then gets reassigned, then UdsTransportProtocolMgr::ChannelReestablished() has to be called.
---------------	---

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.3.2.6 uds\_transport::UdsTransportProtocolHandler::Start

[SWS\_DM\_00322]{DRAFT} **Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::Start** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolHandler</a>
<b>Symbol:</b>	Start()
<b>Syntax:</b>	virtual void Start ()=0;
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	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.

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.3.2.7 uds\_transport::UdsTransportProtocolHandler::Stop

[SWS\_DM\_00323]{DRAFT} **Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::Stop** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolHandler</a>
<b>Symbol:</b>	Stop()





<b>Syntax:</b>	<code>virtual void Stop ()=0;</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	<p>Method to indicate that this UdsTransportProtocolHandler should terminate.</p> <p>If UdsTransportProtocolHandler has stopped, it shall call UdsTransportProtocolMgr::Handler Stopped(UdsTransportProtocolHandlerID)</p> <p>After return from Stop(), the handler-plugin shall NOT call to UdsTransportProtocolMgr with any other method but UdsTransportProtocolMgr::HandlerStopped()</p>

|(RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.3.2.8 uds\_transport::UdsTransportProtocolHandler::Transmit

[SWS\_DM\_00327]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::Transmit [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	Transmit(UdsMessageConstPtr message, ChannelID channelId)	
<b>Syntax:</b>	virtual void Transmit (UdsMessageConstPtr message, ChannelID channel Id)=0;	
<b>Parameters (in):</b>	message	The message to be transmitted as a UdsMessage::Ptr (unique_ptr style). UdsTransportProtocolHandler has to give back this Uds Message::Ptr via UdsTransportProtocolMgr::TransmitConfirmation() to signal, that it is done with this message.
	channelId	identification of channel on which to transmit.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	<p>Transmit a Uds message via the underlying Uds Transport Protocol channel.</p> <p>This transmit API covers T_Data.req of ISO 14229-2 Figure 2.</p>	

|(RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.3.2.9 uds\_transport::UdsTransportProtocolHandler::GetPeriodicHandler

[SWS\_DM\_01068]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolHandler::GetPeriodicHandler [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolHandler	
<b>Symbol:</b>	GetPeriodicHandler()	
<b>Syntax:</b>	virtual ara::core::Result< UdsTransportProtocolPeriodicHandler & > Get PeriodicHandler ();	



△

<b>Return value:</b>	ara::core::Result< UdsTransportProtocolPeriodicHandler & >	UdsTransportProtocolPeriodicHandler reference if periodic transmissions are supported on this transport protocol, an error if not supported
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Returns the corresponding periodic TP handler.	

]([RS\\_Diag\\_04215](#))

### B.1.3.2.10 uds\_transport::UdsTransportProtocolHandler::transportProtocolManager variable

[SWS\_DM\_09025]{DRAFT} **Definition of API variable ara::diag::uds\_transport::UdsTransportProtocolHandler::transportProtocolManager** [

<b>Kind:</b>	variable
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_handler.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolHandler</a>
<b>Symbol:</b>	transportProtocolManager
<b>Type:</b>	UdsTransportProtocolMgr &
<b>Syntax:</b>	UdsTransportProtocolMgr& transportProtocolManager;
<b>Description:</b>	The UdsTransportProtocolMgr used/provided by the DM/DCM.
<b>Visibility:</b>	protected

]()

### B.1.4 UdsTransportProtocolMgr Class

[SWS\_DM\_00306]{DRAFT} **Definition of API class ara::diag::uds\_transport::UdsTransportProtocolMgr** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsTransportProtocolMgr
<b>Syntax:</b>	class UdsTransportProtocolMgr {...};
<b>Description:</b>	–

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

## B.1.4.1 UdsTransportProtocolMgrClass Types

### B.1.4.1.1 uds\_transport::UdsTransportProtocolMgr::GlobalChannelIdentifier

[SWS\_DM\_09021]{DRAFT} Definition of API type ara::diag::uds\_transport::UdsTransportProtocolMgr::GlobalChannelIdentifier [

<b>Kind:</b>	type alias
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolMgr</a>
<b>Symbol:</b>	GlobalChannelIdentifier
<b>Syntax:</b>	using GlobalChannelIdentifier = std::tuple<UdsTransportProtocolHandlerID, ChannelID>;
<b>Description:</b>	Type of tuple to pack UdsTransportProtocolHandlerID and ChannelID together, to form a global unique (among all used UdsTransportProtocolHandlers within DM) identifier of a UdsTransport Protocol channel.

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.4.1.2 uds\_transport::UdsTransportProtocolMgr::IndicationResult

[SWS\_DM\_00384]{DRAFT} Definition of API enum ara::diag::uds\_transport::UdsTransportProtocolMgr::IndicationResult [

<b>Kind:</b>	enumeration								
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"								
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"								
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolMgr</a>								
<b>Symbol:</b>	IndicationResult								
<b>Underlying type:</b>	std::uint8_t								
<b>Syntax:</b>	enum class IndicationResult : std::uint8_t {...};								
<b>Values:</b>	<table border="1"> <tr> <td>kIndicationOk= 0</td> <td>–</td> </tr> <tr> <td>kIndicationOccupied= 1</td> <td>–</td> </tr> <tr> <td>kIndicationOverflow= 2</td> <td>–</td> </tr> <tr> <td>kIndicationUnknown TargetAddress= 3</td> <td>–</td> </tr> </table>	kIndicationOk= 0	–	kIndicationOccupied= 1	–	kIndicationOverflow= 2	–	kIndicationUnknown TargetAddress= 3	–
kIndicationOk= 0	–								
kIndicationOccupied= 1	–								
kIndicationOverflow= 2	–								
kIndicationUnknown TargetAddress= 3	–								
<b>Description:</b>	Allowed return values from the DM Server to the Transport Protocol Handler Instance for the received IndicateMessage.								

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.4.1.3 uds\_transport::UdsTransportProtocolMgr::TransmissionResult

[SWS\_DM\_00307]{DRAFT} Definition of API enum ara::diag::uds\_transport::UdsTransportProtocolMgr::TransmissionResult [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	TransmissionResult	
<b>Underlying type:</b>	std::uint8_t	
<b>Syntax:</b>	enum class TransmissionResult : std::uint8_t {...};	
<b>Values:</b>	kTransmitOk= 0	–
	kTransmitFailed= 1	–
<b>Description:</b>	Values denoting the Transmission Confirmation Result provided by the Transport protocol Handler Instance to the DM Server.	

]([RS\\_Diag\\_04172](#), [RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.4.2 UdsTransportProtocolMgr Methods

#### B.1.4.2.1 uds\_transport::UdsTransportProtocolMgr::~~UdsTransportProtocolMgr

[SWS\_DM\_01524]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::~~UdsTransportProtocolMgr [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	~UdsTransportProtocolMgr()	
<b>Syntax:</b>	virtual ~UdsTransportProtocolMgr () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructor of class UdsTransportProtocolMgr.	

]([RS\\_AP\\_00134](#))



### B.1.4.2.2 uds\_transport::UdsTransportProtocolMgr::ChannelReestablished

[SWS\_DM\_00313]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::ChannelReestablished [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	ChannelReestablished(GlobalChannelIdentifier globalChannelId)	
<b>Syntax:</b>	virtual void ChannelReestablished (GlobalChannelIdentifier globalChannelId)=0;	
<b>Parameters (in):</b>	globalChannelId	transport protocol channel, which is available again.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	notification call from the given transport channel, that it has been reestablished since the last (Re)Start from the UdsTransportProtocolHandler to which this channel belongs. To activate this notification a previous call to UdsTransportProtocolHandler::NotifyReestablishment() has to be done. See further documentation at UdsTransportProtocolHandler::NotifyReestablishment().	

]([RS\\_AP\\_00120](#), [RS\\_AP\\_00121](#), [RS\\_AP\\_00122](#), [RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.4.2.3 uds\_transport::UdsTransportProtocolMgr::HandleMessage

[SWS\_DM\_00311]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::HandleMessage [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	HandleMessage(UdsMessagePtr message)	
<b>Syntax:</b>	virtual void HandleMessage (UdsMessagePtr message)=0;	
<b>Parameters (in):</b>	message	The Uds message ptr (unique_ptr semantics) with the request. Ownership of the UdsMessage is given back to the generic DM core here.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Hands over a valid received Uds message (currently this is only a request type) from transport layer to session layer. It corresponds to T_Data.ind of Figure 2 from ISO 14229-2. The behavior is asynchronously. I.e. the UdsMessage is handed over to Session Layer and it is expected, that it "instantly" returns, which means, that real processing of the message shall be done asynchronously!	

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

### B.1.4.2.4 uds\_transport::UdsTransportProtocolMgr::HandlerStopped

[SWS\_DM\_00314]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::HandlerStopped [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	HandlerStopped(UdsTransportProtocolHandlerId handlerId)	
<b>Syntax:</b>	virtual void HandlerStopped (UdsTransportProtocolHandlerID handlerId)=0;	
<b>Parameters (in):</b>	handlerId	indication, which plugin stopped.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	notification from handler, that it has stopped now (e.g. closed down network connections, freed resources, etc...)  This callback is expected as a reaction from handler to a call to UdsTransportProtocolHandler::Stop.	

](RS\_AP\_00120, RS\_AP\_00121, RS\_AP\_00122, RS\_Diag\_04147, RS\_Diag\_04168)

### B.1.4.2.5 uds\_transport::UdsTransportProtocolMgr::IndicateMessage

[SWS\_DM\_00309]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::IndicateMessage [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	IndicateMessage(UdsMessage::Address sourceAddr, UdsMessage::Address targetAddr, UdsMessage::TargetAddressType type, GlobalChannelIdentifier globalChannelId, std::size_t size, Priority priority, ProtocolKind protocolKind, ara::core::Span< const std::uint8_t > payloadInfo)	
<b>Syntax:</b>	virtual std::pair< IndicationResult, UdsMessagePtr > IndicateMessage ( UdsMessage::Address sourceAddr, UdsMessage::Address targetAddr, UdsMessage::TargetAddressType type, GlobalChannelIdentifier globalChannelId, std::size_t size, Priority priority, ProtocolKind protocolKind, ara::core::Span< const std::uint8_t > payloadInfo)=0;	
<b>Parameters (in):</b>	sourceAddr	UDS source address of message
	targetAddr	UDS target address of message
	type	indication whether its is phys/func request
	globalChannelId	transport protocol channel on which message start happened
	size	size in bytes of the UdsMessage starting from SID.
	priority	the priority of the given message, used for prioritization of conversations.
	protocolKind	identifier of protocol kind associated to message
	payloadInfo	View onto the first received payload bytes, if any. This view shall be used only within this function call. It is recommended that the TP provides at least the first two bytes of the request message, so the DM can identify a functional TesterPresent.





<b>Return value:</b>	std::pair< Indication Result, UdsMessagePtr >	Pair of IndicationResult and a pointer to UdsMessage owned/created by DM core and returned to the handler to get filled.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Indicates a message start. This is an interface, which is just served/called by UdsTransport ProtocolHandlers, which return true from UdsTransportProtocolHandlers::isStartOfMessage IndicationSupported().	

]([RS\\_AP\\_00120](#), [RS\\_AP\\_00121](#), [RS\\_AP\\_00122](#), [RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

#### B.1.4.2.6 uds\_transport::UdsTransportProtocolMgr::NotifyMessageFailure

[SWS\_DM\_00310]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::NotifyMessageFailure [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	NotifyMessageFailure(UdsMessagePtr message)	
<b>Syntax:</b>	virtual void NotifyMessageFailure (UdsMessagePtr message)=0;	
<b>Parameters (in):</b>	message	the pointer to UdsMessage handed back over to the session layer.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Indicates, that the message indicated via IndicateMessage() has failure and will not lead to a final HandleMessage() call.	

]([RS\\_Diag\\_04147](#), [RS\\_Diag\\_04168](#))

#### B.1.4.2.7 uds\_transport::UdsTransportProtocolMgr::TransmitConfirmation

[SWS\_DM\_00312]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::TransmitConfirmation [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	class ara::diag::uds_transport::UdsTransportProtocolMgr	
<b>Symbol:</b>	TransmitConfirmation(UdsMessageConstPtr message, TransmissionResult result)	
<b>Syntax:</b>	virtual void TransmitConfirmation (UdsMessageConstPtr message, TransmissionResult result)=0;	
<b>Parameters (in):</b>	message	for which message (created in IndicateMessage()) this is the confirmation.
	result	Result of transmission. In case UDS message could be transmitted on network layer: kTransmitOk, kTransmitFailed else.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	





<b>Description:</b>	notification about the outcome of a transmit request called by core DM at the handler via UdsTransportProtocolHandler::Transmit  This transmit API covers T_Data.con of ISO 14229-2 Figure 2.
---------------------	---

](RS\_Diag\_04172, RS\_Diag\_04147, RS\_Diag\_04168)

#### B.1.4.2.8 uds\_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation

[SWS\_DM\_01069]{DRAFT} Definition of API function ara::diag::uds\_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_mgr.h"	
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolMgr</a>	
<b>Symbol:</b>	PeriodicTransmitConfirmation(ara::core::Vector< UdsMessageConstPtr > messages, std::size_t numberOfSentMessages)	
<b>Syntax:</b>	virtual void PeriodicTransmitConfirmation (ara::core::Vector< <a href="#">UdsMessageConstPtr</a> > messages, std::size_t numberOfSentMessages)=0;	
<b>Parameters (in):</b>	messages	The same ordered list of messages previously passed to UdsTransportProtocolPeriodicHandler::PeriodicTransmit.
	numberOfSentMessages	The number of successfully sent messages from the "messages" list.
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Confirmation of sent messages and number.	

](RS\_Diag\_04215)

#### B.1.5 UdsTransportProtocolPeriodicHandler Class

[SWS\_DM\_01064]{DRAFT} Definition of API class ara::diag::uds\_transport::UdsTransportProtocolPeriodicHandler [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_periodic_handler.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag::uds_transport
<b>Symbol:</b>	UdsTransportProtocolPeriodicHandler
<b>Syntax:</b>	class UdsTransportProtocolPeriodicHandler {...};
<b>Description:</b>	UdsTransportProtocolPeriodicHandler class to support 0x2A service from ISO.

](RS\_Diag\_04215)

## B.1.5.1 UdsTransportProtocolPeriodicHandler Methods

### B.1.5.1.1 uds\_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages

[SWS\_DM\_01065]{DRAFT} Definition of API function `ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler</a>	
<b>Symbol:</b>	GetNumberOfPeriodicMessages(ChannelID channelId)	
<b>Syntax:</b>	virtual std::size_t GetNumberOfPeriodicMessages (ChannelID channelId) const =0;	
<b>Parameters (in):</b>	channelId	The concrete connection to send over
<b>Return value:</b>	std::size_t	number of periodic messages
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Reports the TP implementation and connection specific number of periodic messages.	

]([RS\\_Diag\\_04215](#))

### B.1.5.1.2 uds\_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength

[SWS\_DM\_01066]{DRAFT} Definition of API function `ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler::GetMaxPayloadLength` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
<b>Scope:</b>	<a href="#">class ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler</a>	
<b>Symbol:</b>	GetMaxPayloadLength(ChannelID channelId)	
<b>Syntax:</b>	virtual std::size_t GetMaxPayloadLength (ChannelID channelId) const =0;	
<b>Parameters (in):</b>	channelId	The concrete connection for which the maximum payload length is reported.
<b>Return value:</b>	std::size_t	supported payload length
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Reports the maximum payload length supported for a single periodic transmission on the channel.	

]([RS\\_Diag\\_04215](#))

### B.1.5.1.3 uds\_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit

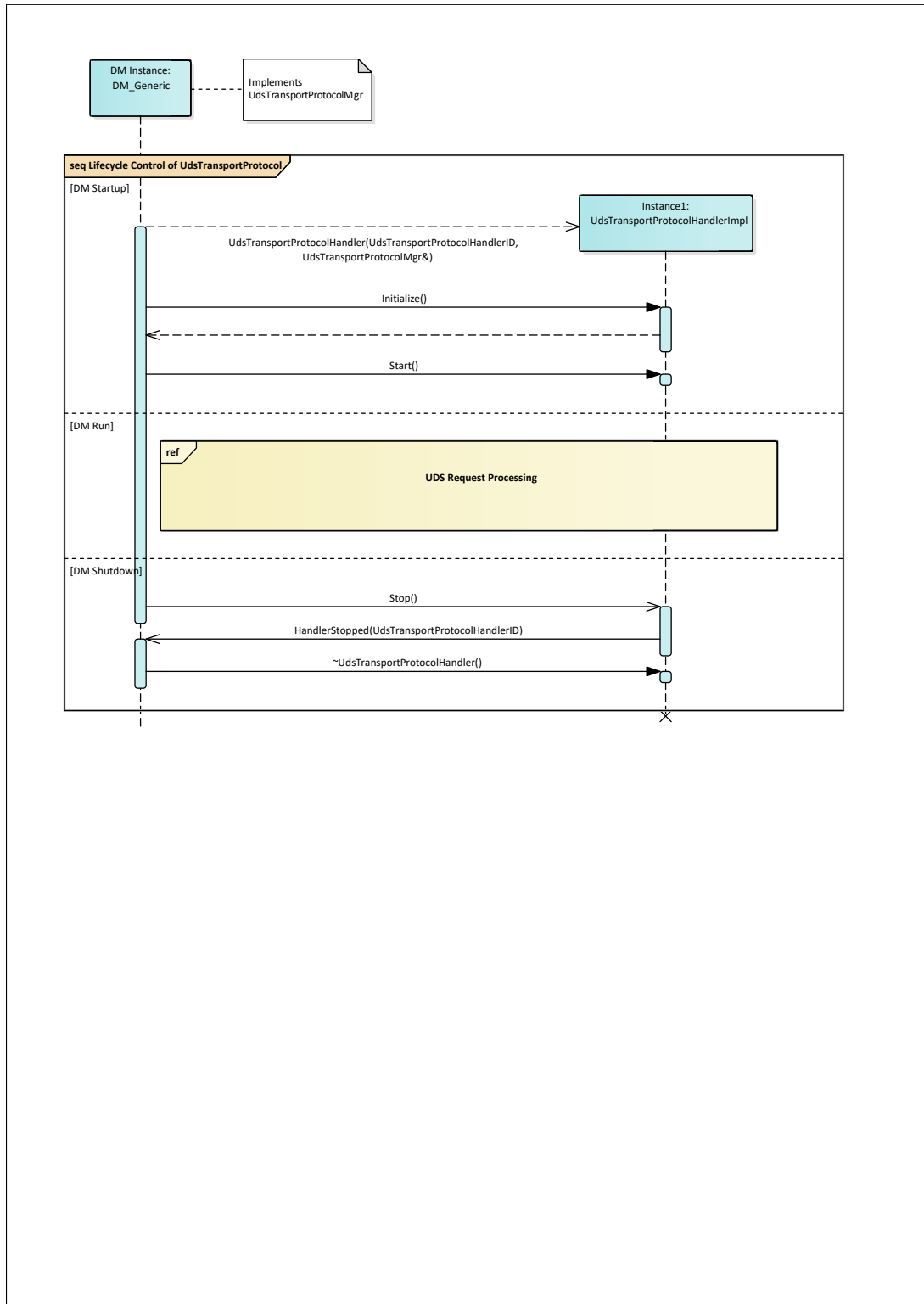
[SWS\_DM\_01067]{DRAFT} Definition of API function `ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler::PeriodicTransmit` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
<b>Scope:</b>	class <code>ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler</code>	
<b>Symbol:</b>	PeriodicTransmit(ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages)	
<b>Syntax:</b>	virtual void PeriodicTransmit (ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages)=0;	
<b>Parameters (in):</b>	channelId	The concrete connection to send over
	messages	Ordered list of messages to send at once
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Sends all the messages in the list in the given order. If one message transmission fails, the send process is stopped and the PeriodicTransmitConfirmation is invoked with the number of sent messages and the same Vector with UdsMessages.	

](RS\_Diag\_04215)

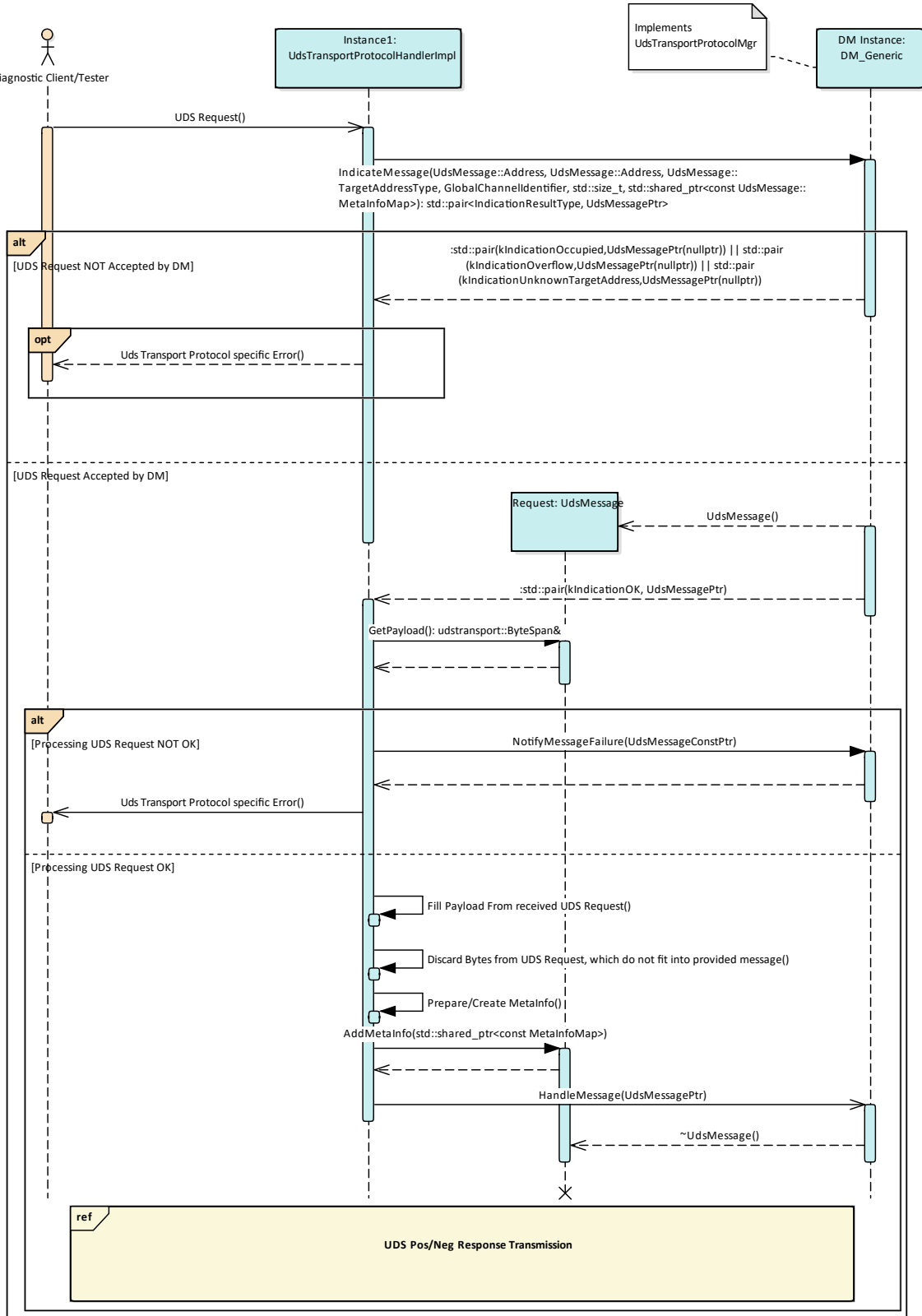
## B.1.6 Sequence Diagrams of UDS Transport Layer Interaction

### B.1.6.1 Lifecycle



**Figure B.1: UDS Transport Lifecycle**

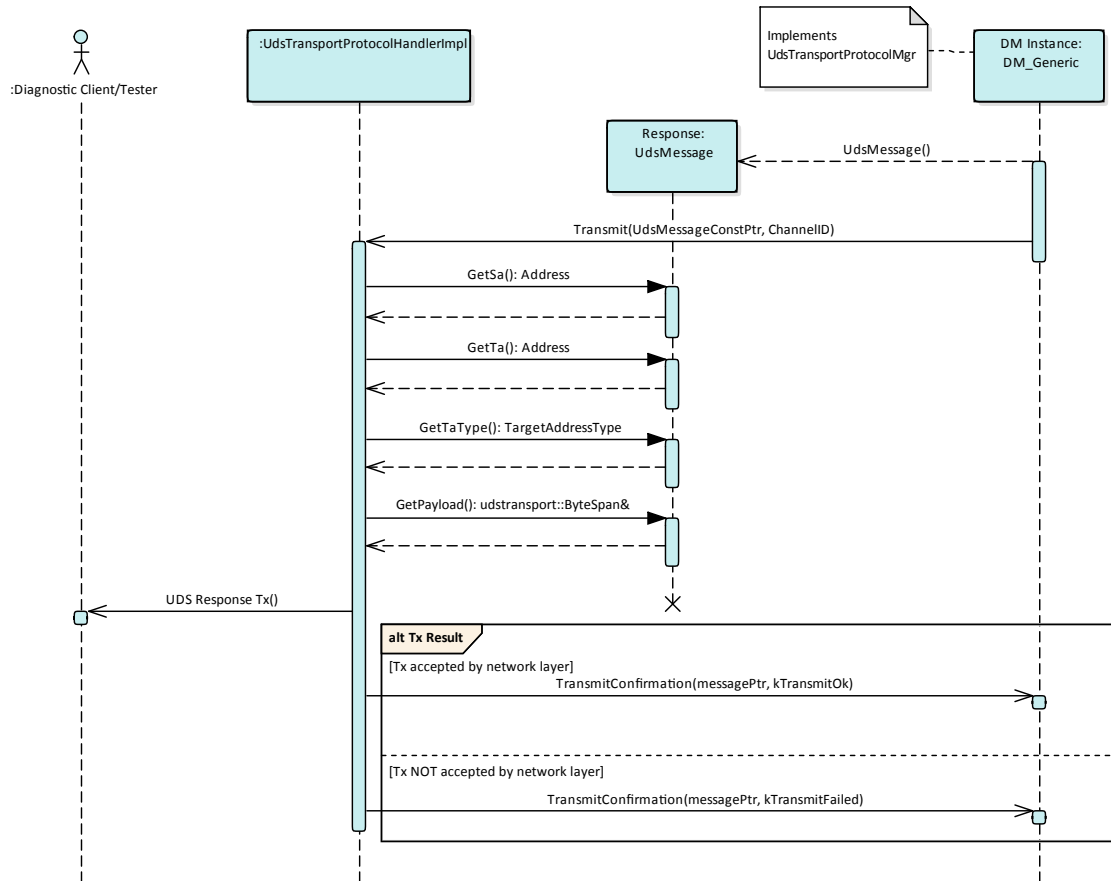
**B.1.6.2 UDS Request Processing**



**Figure B.2: UDS Transport Request Processing**

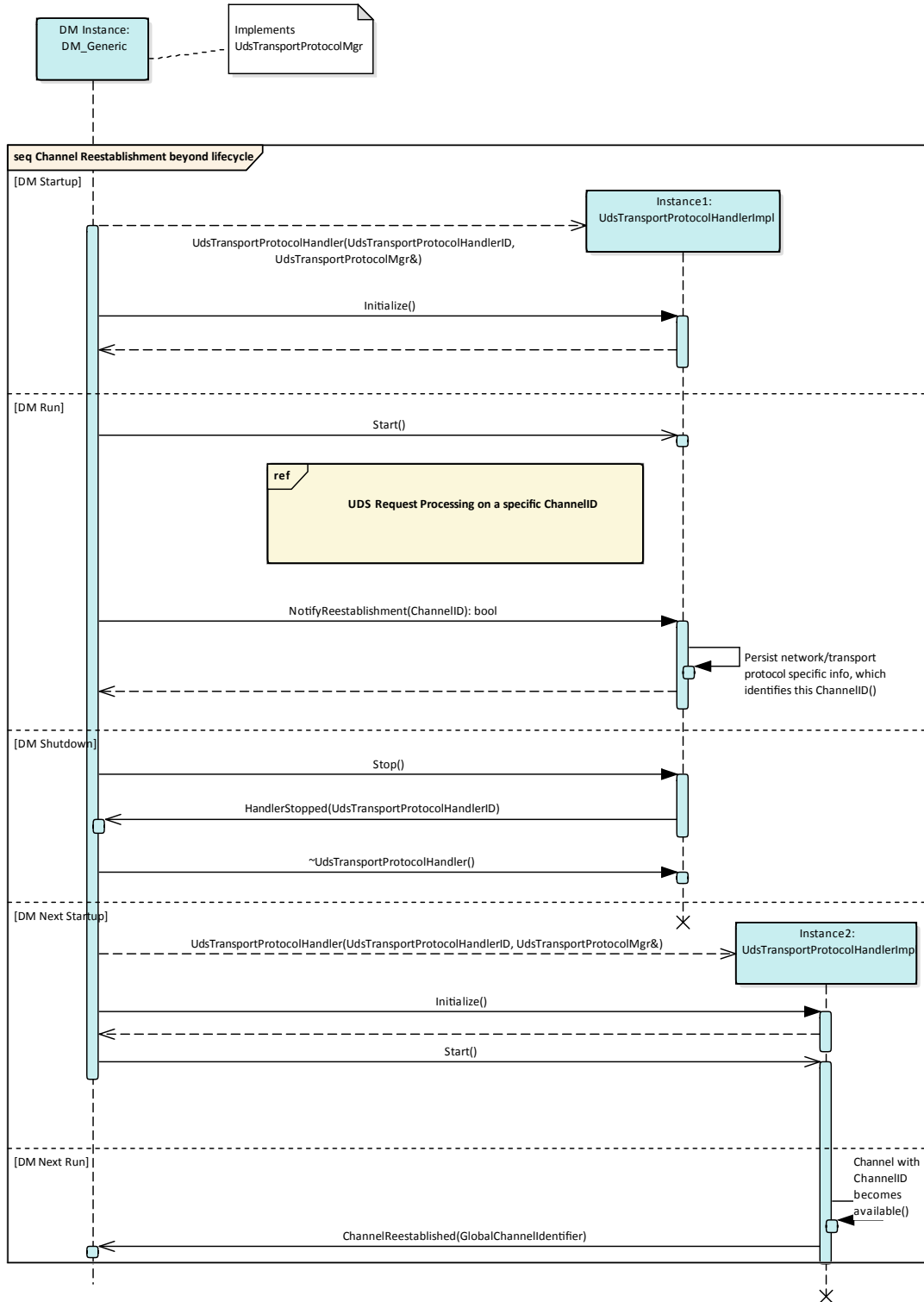


**B.1.6.3 UDS Response Transmission**



**Figure B.3: UDS Response Transmission**

**B.1.6.4 Channel Reestablishment**



**Figure B.4: UDS Transport Channel Reestablishment**

## B.2 C++ DoIP API Interfaces

### B.2.1 DoIPGroupIdentification class

The *InstanceSpecifier* is only compatible with *PortInterface* of *Diagnostic-DoIPGroupIdentificationInterface*.

**[SWS\_DM\_00720]{DRAFT} Definition of API class ara::diag::DoIPGroupIdentification** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DoIPGroupIdentification
<b>Syntax:</b>	class DoIPGroupIdentification {...};
<b>Description:</b>	DoIPGroupIdentificationInterface.

]([SRS\\_Eth\\_00026](#))

#### B.2.1.1 diag::DoIPGroupIdentification::DoIPGroupIdentificationType type

**[SWS\_DM\_00721]{DRAFT} Definition of API class ara::diag::DoIPGroupIdentification::GidStatus** [

<b>Kind:</b>	struct
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPGroupIdentification</a>
<b>Symbol:</b>	GidStatus
<b>Syntax:</b>	struct GidStatus {...};
<b>Description:</b>	Response data of positive response message.

]([SRS\\_Eth\\_00026](#))

#### B.2.1.2 diag::DoIPGroupIdentification::DoIPGroupIdentification function

**[SWS\_DM\_00722]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPGroupIdentification</a>





<b>Symbol:</b>	DoIPGroupIdentification(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit DoIPGroupIdentification (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPGroup IdentificationInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of DoIPGroupIdentification.	

]([RS\\_AP\\_00137](#), [SRS\\_Eth\\_00026](#))

### B.2.1.3 diag::DoIPGroupIdentification::DoIPGroupIdentification(DoIPGroupIdentification&& move constructor

[SWS\_DM\_01642]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPGroupIdentification</a>
<b>Symbol:</b>	DoIPGroupIdentification(DoIPGroupIdentification &&)
<b>Syntax:</b>	DoIPGroupIdentification (DoIPGroupIdentification &&) noexcept=delete;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Move constructor of DoIPGroupIdentification.

]([RS\\_AP\\_00147](#))

### B.2.1.4 diag::DoIPGroupIdentification::DoIPGroupIdentification(DoIPGroupIdentification&) copy constructor

[SWS\_DM\_01641]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::DoIPGroupIdentification [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPGroupIdentification</a>
<b>Symbol:</b>	DoIPGroupIdentification(DoIPGroupIdentification &)
<b>Syntax:</b>	DoIPGroupIdentification (DoIPGroupIdentification &)=delete;
<b>Description:</b>	DoIPGroupIdentification shall be a single not copy-able instance.

]([RS\\_AP\\_00147](#))

### B.2.1.5 diag::DoIPGroupIdentification::DoIPGroupIdentification& operator=(DoIPGroupIdentification&&) move operator

[SWS\_DM\_01640]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification
<b>Symbol:</b>	operator=(DoIPGroupIdentification &&)
<b>Syntax:</b>	DoIPGroupIdentification & operator= (DoIPGroupIdentification &&)=delete;
<b>Description:</b>	Move assignment operator of DoIPGroupIdentification.

](RS\_AP\_00147)

### B.2.1.6 diag::DoIPGroupIdentification::DoIPGroupIdentification& operator=(DoIPGroupIdentification&) copy operator

[SWS\_DM\_01639]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification
<b>Symbol:</b>	operator=(DoIPGroupIdentification &)
<b>Syntax:</b>	DoIPGroupIdentification & operator= (DoIPGroupIdentification &)=delete;
<b>Description:</b>	DoIPGroupIdentification shall be a single not assignable instance.

](RS\_AP\_00147)

### B.2.1.7 diag::DoIPGroupIdentification::~~DoIPGroupIdentification function

[SWS\_DM\_00723]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::~~DoIPGroupIdentification [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification
<b>Symbol:</b>	~DoIPGroupIdentification()
<b>Syntax:</b>	virtual ~DoIPGroupIdentification () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DoIPGroupIdentification.

](RS\_AP\_00134, SRS\_Eth\_00026)

### B.2.1.8 diag::DoIPGroupIdentification::GetGidStatus function

[SWS\_DM\_00724]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::GetGidStatus [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"	
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification	
<b>Symbol:</b>	GetGidStatus()	
<b>Syntax:</b>	virtual ara::core::Future< GidStatus > GetGidStatus ()=0;	
<b>Return value:</b>	ara::core::Future< GidStatus >	group identification and state
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Called to get the current GID state for the DoIP protocol.	

](RS\_AP\_00138, SRS\_Eth\_00026)

### B.2.1.9 diag::DoIPGroupIdentification::Offer function

[SWS\_DM\_00725]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"	
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

](RS\_AP\_00119, RS\_AP\_00139, SRS\_Eth\_00026)

### B.2.1.10 diag::DoIPGroupIdentification::StopOffer function

[SWS\_DM\_00726]{DRAFT} Definition of API function ara::diag::DoIPGroupIdentification::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_group_identification.h"	
<b>Scope:</b>	class ara::diag::DoIPGroupIdentification	
<b>Symbol:</b>	StopOffer()	





<b>Syntax:</b>	<code>void StopOffer ();</code>
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([SRS\\_Eth\\_00026](#))

## B.2.2 DoIPPowerMode class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [Diagnostic-DoIPPowerModeInterface](#).

### [SWS\_DM\_00731]{DRAFT} Definition of API class `ara::diag::DoIPPowerMode` [

<b>Kind:</b>	class
<b>Header file:</b>	<code>#include "ara/diag/doip_power_mode.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	namespace <code>ara::diag</code>
<b>Symbol:</b>	<code>DoIPPowerMode</code>
<b>Syntax:</b>	<code>class DoIPPowerMode {...};</code>
<b>Description:</b>	<code>DiagnosticDoIPPowerModeInterface</code> .

]([SRS\\_Eth\\_00080](#))

### B.2.2.1 `diag::PowerModeType` type

#### [SWS\_DM\_00730]{DRAFT} Definition of API enum `ara::diag::PowerModeType` [

<b>Kind:</b>	enumeration	
<b>Header file:</b>	<code>#include "ara/diag/doip_power_mode.h"</code>	
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>	
<b>Scope:</b>	namespace <code>ara::diag</code>	
<b>Symbol:</b>	<code>PowerModeType</code>	
<b>Underlying type:</b>	–	
<b>Syntax:</b>	<code>enum class PowerModeType {...};</code>	
<b>Values:</b>	<code>kNotReady= 0x00</code>	not all ECUs accessible via DoIP can communicate
	<code>kReady= 0x01</code>	all ECUs accessible via DoIP can communicate
	<code>kNotSupported= 0x02</code>	the Diagnostic Information Power Mode Information Request message is not supported
<b>Description:</b>	PowerMode as defined in ISO13400-2.	

]([SRS\\_Eth\\_00080](#))

### B.2.2.2 diag::DoIPPowerMode::DoIPPowerMode function

[SWS\_DM\_00732]{DRAFT} **Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>	
<b>Symbol:</b>	DoIPPowerMode(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit DoIPPowerMode (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPPowerModeInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of DoIPPowerMode.	

]([RS\\_AP\\_00137](#), [SRS\\_Eth\\_00080](#))

### B.2.2.3 diag::DoIPPowerMode::DoIPPowerMode(DoIPPowerMode&&) move constructor

[SWS\_DM\_01646]{DRAFT} **Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>	
<b>Symbol:</b>	DoIPPowerMode(DoIPPowerMode &&)	
<b>Syntax:</b>	DoIPPowerMode (DoIPPowerMode &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of DoIPPowerMode.	

]([RS\\_AP\\_00147](#))

### B.2.2.4 diag::DoIPPowerMode::DoIPPowerMode(DoIPPowerMode&) copy constructor

[SWS\_DM\_01645]{DRAFT} **Definition of API function ara::diag::DoIPPowerMode::DoIPPowerMode** [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>	





△

<b>Symbol:</b>	DoIPPowerMode(DoIPPowerMode &)
<b>Syntax:</b>	DoIPPowerMode (DoIPPowerMode &)=delete;
<b>Description:</b>	DoIPPowerMode shall be a single not copy-able instance.

](RS\_AP\_00147)

### B.2.2.5 diag::DoIPPowerMode::DoIPPowerMode&tor=(DoIPPowerMode&&) move operator opera-

[SWS\_DM\_01644]{DRAFT} Definition of API function ara::diag::DoIPPowerMode::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"
<b>Scope:</b>	class ara::diag::DoIPPowerMode
<b>Symbol:</b>	operator=(DoIPPowerMode &&)
<b>Syntax:</b>	DoIPPowerMode & operator= (DoIPPowerMode &&)=delete;
<b>Description:</b>	Move assignment operator of DoIPPowerMode.

](RS\_AP\_00147)

### B.2.2.6 diag::DoIPPowerMode::DoIPPowerMode&tor=(DoIPPowerMode&) copy operator opera-

[SWS\_DM\_01643]{DRAFT} Definition of API function ara::diag::DoIPPowerMode::operator= [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"
<b>Scope:</b>	class ara::diag::DoIPPowerMode
<b>Symbol:</b>	operator=(DoIPPowerMode &)
<b>Syntax:</b>	DoIPPowerMode & operator= (DoIPPowerMode &)=delete;
<b>Description:</b>	DoIPPowerMode shall be a single not assignable instance.

](RS\_AP\_00147)

### B.2.2.7 diag::DoIPPowerMode::~~DoIPPowerMode function

[SWS\_DM\_00733]{DRAFT} Definition of API function ara::diag::DoIPPowerMode::~~DoIPPowerMode [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>
<b>Symbol:</b>	~DoIPPowerMode()
<b>Syntax:</b>	virtual ~DoIPPowerMode () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DoIPPowerMode.

]([RS\\_AP\\_00134](#), [SRS\\_Eth\\_00080](#))

### B.2.2.8 diag::DoIPPowerMode::GetDoIPPowerMode function

[SWS\_DM\_00734]{DRAFT} Definition of API function ara::diag::DoIPPowerMode::GetDoIPPowerMode [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"		
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>		
<b>Symbol:</b>	GetDoIPPowerMode()		
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">PowerModeType</a> > GetDoIPPowerMode ()=0;		
<b>Return value:</b>	<table border="1"> <tr> <td>ara::core::Future&lt; Power ModeType &gt;</td> <td>current diagnostic power mode</td> </tr> </table>	ara::core::Future< Power ModeType >	current diagnostic power mode
ara::core::Future< Power ModeType >	current diagnostic power mode		
<b>Exception Safety:</b>	not exception safe		
<b>Description:</b>	Called to get the current Power Mode for the DoIP protocol.		

]([RS\\_AP\\_00138](#), [SRS\\_Eth\\_00080](#))

### B.2.2.9 diag::DoIPPowerMode::Offer function

[SWS\_DM\_00735]{DRAFT} Definition of API function ara::diag::DoIPPowerMode::Offer [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"		
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>		
<b>Symbol:</b>	Offer()		
<b>Syntax:</b>	ara::core::Result< void > Offer ();		
<b>Return value:</b>	<table border="1"> <tr> <td>ara::core::Result&lt; void &gt;</td> <td>-</td> </tr> </table>	ara::core::Result< void >	-
ara::core::Result< void >	-		
<b>Exception Safety:</b>	not exception safe		





<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [SRS\\_Eth\\_00080](#))

### B.2.2.10 diag::DoIPPowerMode::StopOffer function

[SWS\_DM\_00736]{DRAFT} **Definition of API function ara::diag::DoIPPowerMode::StopOffer** [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_power_mode.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPPowerMode</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.

]([SRS\\_Eth\\_00080](#))

### B.2.3 DoIPActivationLine class

The *InstanceSpecifier* is only compatible with PortInterface of [DiagnosticDoIPActivationLineInterface](#). Note : For DoIPActivationLineInterface, DM has to have a R-PORT.

[SWS\_DM\_00830]{DRAFT} **Definition of API class ara::diag::DoIPActivationLine** [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DoIPActivationLine
<b>Syntax:</b>	class DoIPActivationLine {...};
<b>Description:</b>	DiagnosticDoIPActivationLineInterface.

]([RS\\_Diag\\_04242](#))

### B.2.3.1 diag::DoIPActivationLine::DoIPActivationLine function

[SWS\_DM\_00831]{DRAFT} Definition of API function `ara::diag::DoIPActivationLine::DoIPActivationLine` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>	
<b>Symbol:</b>	DoIPActivationLine(const ara::core::InstanceSpecifier &specifier)	
<b>Syntax:</b>	explicit DoIPActivationLine (const ara::core::InstanceSpecifier &specifier);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPActivationLineInterface
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of DoIPActivationLine.	

]([RS\\_Diag\\_04242](#))

### B.2.3.2 diag::DoIPActivationLine::DoIPActivationLine(DoIPActivationLine&&) move constructor

[SWS\_DM\_01634]{DRAFT} Definition of API function `ara::diag::DoIPActivationLine::DoIPActivationLine` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>	
<b>Symbol:</b>	DoIPActivationLine(DoIPActivationLine &&)	
<b>Syntax:</b>	DoIPActivationLine (DoIPActivationLine &&) noexcept=delete;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Move constructor of DoIPActivationLine.	

]([RS\\_AP\\_00147](#))

### B.2.3.3 diag::DoIPActivationLine::DoIPActivationLine(DoIPActivationLine&) copy constructor

[SWS\_DM\_01633]{DRAFT} Definition of API function `ara::diag::DoIPActivationLine::DoIPActivationLine` [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>	
<b>Symbol:</b>	DoIPActivationLine(DoIPActivationLine &)	



△

<b>Syntax:</b>	<code>DoIPActivationLine (DoIPActivationLine &amp;)=delete;</code>
<b>Description:</b>	DoIPActivationLine shall be a single not copy-able instance.

](RS\_AP\_00147)

#### B.2.3.4 `diag::DoIPActivationLine::DoIPActivationLine&` opera- `tor=(DoIPActivationLine&&) move operator`

[SWS\_DM\_01632][DRAFT] Definition of API function `ara::diag::DoIPActivationLine::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/doip_activationline.h"</code>
<b>Scope:</b>	<code>class ara::diag::DoIPActivationLine</code>
<b>Symbol:</b>	<code>operator=(DoIPActivationLine &amp;&amp;)</code>
<b>Syntax:</b>	<code>DoIPActivationLine &amp; operator= (DoIPActivationLine &amp;&amp;)=delete;</code>
<b>Description:</b>	Move assignment operator of DoIPActivationLine.

](RS\_AP\_00147)

#### B.2.3.5 `diag::DoIPActivationLine::DoIPActivationLine&` opera- `tor=(DoIPActivationLine&) copy operator`

[SWS\_DM\_01631][DRAFT] Definition of API function `ara::diag::DoIPActivationLine::operator=` [

<b>Kind:</b>	function
<b>Header file:</b>	<code>#include "ara/diag/doip_activationline.h"</code>
<b>Scope:</b>	<code>class ara::diag::DoIPActivationLine</code>
<b>Symbol:</b>	<code>operator=(DoIPActivationLine &amp;)</code>
<b>Syntax:</b>	<code>DoIPActivationLine &amp; operator= (DoIPActivationLine &amp;)=delete;</code>
<b>Description:</b>	DoIPActivationLine shall be a single not assignable instance.

](RS\_AP\_00147)

### B.2.3.6 diag::DoIPActivationLine::~DoIPActivationLine function

[SWS\_DM\_00832]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::~DoIPActivationLine [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>
<b>Symbol:</b>	~DoIPActivationLine()
<b>Syntax:</b>	virtual ~DoIPActivationLine () noexcept=default;
<b>Exception Safety:</b>	noexcept
<b>Description:</b>	Destructor of DoIPActivationLine.

]([RS\\_Diag\\_04242](#), [RS\\_AP\\_00134](#))

### B.2.3.7 diag::DoIPActivationLine::GetNetworkInterfaceId function

[SWS\_DM\_00833]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::GetNetworkInterfaceId [

<b>Kind:</b>	function		
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"		
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>		
<b>Symbol:</b>	GetNetworkInterfaceId()		
<b>Syntax:</b>	virtual ara::core::Future< std::uint8_t > GetNetworkInterfaceId ()=0;		
<b>Return value:</b>	<table border="1"> <tr> <td>ara::core::Future&lt; std::uint8_t &gt;</td> <td>network interface id for which this activation line is responsible.</td> </tr> </table>	ara::core::Future< std::uint8_t >	network interface id for which this activation line is responsible.
ara::core::Future< std::uint8_t >	network interface id for which this activation line is responsible.		
<b>Exception Safety:</b>	not exception safe		
<b>Description:</b>	Called to get the network interface Id (see DolpNetworkConfiguration.networkInterfaceId) for which this DoIPActivationLine instance is responsible.		
<b>Notes:</b>	If the reported DolpNetworkConfiguration.networkInterfaceId belongs to a DolpNetwork Configuration with property isActivationLineDependent = 'FALSE', this is an error!		

]([RS\\_Diag\\_04242](#))

### B.2.3.8 diag::DoIPActivationLine::UpdateActivationLineState function

[SWS\_DM\_00834]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::UpdateActivationLineState [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>
<b>Symbol:</b>	UpdateActivationLineState(bool)
<b>Syntax:</b>	virtual void UpdateActivationLineState (bool)=0;



△

<b>DIRECTION NOT DEFINED</b>	bool	-
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Called to update current activation line state.	

 ]([RS\\_Diag\\_04242](#))

### B.2.3.9 diag::DoIPActivationLine::GetActivationLineState function

[SWS\_DM\_00835]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::GetActivationLineState [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>	
<b>Symbol:</b>	GetActivationLineState()	
<b>Syntax:</b>	virtual ara::core::Future< bool > GetActivationLineState ()=0;	
<b>Return value:</b>	ara::core::Future< bool >	TRUE in case the activation line is active, else FALSE.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Called to get the current activation line state.	

 ]([RS\\_Diag\\_04242](#))

### B.2.3.10 diag::DoIPActivationLine::Offer function

[SWS\_DM\_00836]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to listen to activation line state changes for the given interface.	

 ]([RS\\_Diag\\_04242](#), [RS\\_AP\\_00119](#))

### B.2.3.11 diag::DoIPActivationLine::StopOffer function

[SWS\_DM\_00837]{DRAFT} Definition of API function ara::diag::DoIPActivationLine::StopOffer [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_activationline.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPActivationLine</a>
<b>Symbol:</b>	StopOffer()
<b>Syntax:</b>	void StopOffer ();
<b>Return value:</b>	None
<b>Exception Safety:</b>	not exception safe
<b>Description:</b>	This StopOffer will disable the provision of activation line state to DM.

]([RS\\_Diag\\_04242](#))

### B.2.4 DoIPTriggerVehicleAnnouncement class

For [DiagnosticDoIPTriggerVehicleAnnouncementInterface](#), DM has to provide a P-Port per supported DoIP network interface.

[SWS\_DM\_00820]{DRAFT} Definition of API class ara::diag::DoIPTriggerVehicleAnnouncement [

<b>Kind:</b>	class
<b>Header file:</b>	#include "ara/diag/doip_trigger_announcement.h"
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"
<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DoIPTriggerVehicleAnnouncement
<b>Syntax:</b>	class DoIPTriggerVehicleAnnouncement {...};
<b>Description:</b>	DiagnosticDoIPTriggerVehicleAnnouncement.

]([RS\\_Diag\\_04242](#))

#### B.2.4.1 diag::DoIPTriggerVehicleAnnouncement::GetDoIPTriggerVehicleAnnouncement function

[SWS\_DM\_00821]{DRAFT} Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::GetDoIPTriggerVehicleAnnouncement [

<b>Kind:</b>	function
<b>Header file:</b>	#include "ara/diag/doip_trigger_announcement.h"
<b>Scope:</b>	<a href="#">class ara::diag::DoIPTriggerVehicleAnnouncement</a>
<b>Symbol:</b>	GetDoIPTriggerVehicleAnnouncement()







<b>Syntax:</b>	static ara::core::Result< DoIPTriggerVehicleAnnouncement & > GetDoIPTriggerVehicleAnnouncement ();	
<b>Return value:</b>	ara::core::Result< DoIPTriggerVehicleAnnouncement & >	DoIPTriggerVehicleAnnouncement object
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Get DoIPTriggerVehicleAnnouncement interface from DM.	

](RS\_Diag\_04242)

### B.2.4.2 diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement function

[SWS\_DM\_00822]{DRAFT} Definition of API function ara::diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_trigger_announcement.h"	
<b>Scope:</b>	class ara::diag::DoIPTriggerVehicleAnnouncement	
<b>Symbol:</b>	TriggerVehicleAnnouncement(std::uint8_t networkInterfaceId)	
<b>Syntax:</b>	virtual ara::core::Result< void > TriggerVehicleAnnouncement (std::uint8_t networkInterfaceId)=0;	
<b>DIRECTION NOT DEFINED</b>	networkInterfaceId	-
<b>Return value:</b>	ara::core::Result< void >	-
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Called by application to trigger DM sending out vehicle announcements on the given network interface Id.	
<b>Notes:</b>	If the reported DolpNetworkConfiguration.networkInterfaceId belongs to a DolpNetwork 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.	

](RS\_Diag\_04242)

### B.2.5 DoIPEntityIdentification class

The *InstanceSpecifier* is only compatible with *PortInterface* of *Diagnostic-DoIPEntityIdentificationInterface*.

[SWS\_DM\_01362]{DRAFT} Definition of API class ara::diag::DoIPEntityIdentification [

<b>Kind:</b>	class	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Forwarding header file:</b>	#include "ara/diag/diag_fwd.h"	





<b>Scope:</b>	namespace ara::diag
<b>Symbol:</b>	DoIPEntityIdentification
<b>Syntax:</b>	<code>class DoIPEntityIdentification {...};</code>
<b>Description:</b>	DoIPEntityIdentification class to obtain EID if required.

]([SRS\\_Eth\\_00026](#))

### B.2.5.1 diag::DoIPEntityIdentification::DoIPEntityIdentificationType type

[SWS\_DM\_01363]{DRAFT} Definition of API class ara::diag::DoIPEntityIdentification::EntityId [

<b>Kind:</b>	struct
<b>Header file:</b>	<code>#include "ara/diag/doip_entity_identification.h"</code>
<b>Forwarding header file:</b>	<code>#include "ara/diag/diag_fwd.h"</code>
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>
<b>Symbol:</b>	EntityId
<b>Syntax:</b>	<code>struct EntityId {...};</code>
<b>Description:</b>	Entity Identification value as defined in ISO13400-2.

]([SRS\\_Eth\\_00026](#))

#### B.2.5.1.1 diag::DoIPEntityIdentification::EntityId entityIdentification

[SWS\_DM\_01496]{DRAFT} Definition of API variable ara::diag::DoIPEntityIdentification::EntityId::entityIdentification [

<b>Kind:</b>	variable
<b>Header file:</b>	<code>#include "ara/diag/doip_entity_identification.h"</code>
<b>Scope:</b>	<a href="#">struct ara::diag::DoIPEntityIdentification::EntityId</a>
<b>Symbol:</b>	entityIdentification
<b>Type:</b>	<code>ara::core::Array&lt; std::uint8_t, 6 &gt;</code>
<b>Syntax:</b>	<code>ara::core::Array&lt;std::uint8_t, 6&gt; entityIdentification;</code>
<b>Description:</b>	Entity Identification value.

]([SRS\\_Eth\\_00026](#))

### B.2.5.2 diag::DoIPEntityIdentification::DoIPEntityIdentification constructor

[SWS\_DM\_01364]{DRAFT} Definition of API function ara::diag::DoIPEntityIdentification::DoIPEntityIdentification [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>	
<b>Symbol:</b>	DoIPEntityIdentification(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
<b>Syntax:</b>	explicit DoIPEntityIdentification (const ara::core::InstanceSpecifier &specifier, <a href="#">ReentrancyType</a> reentrancyType);	
<b>Parameters (in):</b>	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPEntityIdentificationInterface.
	reentrancyType	specifies if interface is callable fully- or non-reentrant.
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Constructor of DoIPEntityIdentification.	

]([RS\\_AP\\_00137](#), [SRS\\_Eth\\_00026](#))

### B.2.5.3 diag::DoIPEntityIdentification::~~DoIPEntityIdentification destructor

[SWS\_DM\_01365]{DRAFT} Definition of API function ara::diag::DoIPEntityIdentification::~~DoIPEntityIdentification [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>	
<b>Symbol:</b>	~DoIPEntityIdentification()	
<b>Syntax:</b>	virtual ~DoIPEntityIdentification () noexcept=default;	
<b>Exception Safety:</b>	noexcept	
<b>Description:</b>	Destructor of DoIPEntityIdentification.	

]([RS\\_AP\\_00134](#), [SRS\\_Eth\\_00026](#))

### B.2.5.4 diag::DoIPEntityIdentification::GetEntityId function

[SWS\_DM\_01366]{DRAFT} Definition of API function ara::diag::DoIPEntityIdentification::GetEntityId [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>	
<b>Symbol:</b>	GetEntityId()	
<b>Syntax:</b>	virtual ara::core::Future< <a href="#">EntityId</a> > GetEntityId ()=0;	





<b>Return value:</b>	ara::core::Future< Entity Id >	Entity identification
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	Called to get the EID for the DoIP protocol.	

]([RS\\_AP\\_00138](#), [SRS\\_Eth\\_00026](#))

### B.2.5.5 diag::DoIPEntityIdentification::Offer function

[SWS\_DM\_01367]{DRAFT} Definition of API function ara::diag::DoIPEntityIdentification::Offer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>	
<b>Symbol:</b>	Offer()	
<b>Syntax:</b>	ara::core::Result< void > Offer ();	
<b>Return value:</b>	ara::core::Result< void >	–
<b>Exception Safety:</b>	not exception safe	
<b>Errors:</b>	DiagErrorDomain::Diag OfferErrc::kAlready Offered	This service was already offered.
<b>Description:</b>	This Offer will enable the DM to forward request messages to this handler.	

]([RS\\_AP\\_00119](#), [RS\\_AP\\_00139](#), [SRS\\_Eth\\_00026](#))

### B.2.5.6 diag::DoIPEntityIdentification::StopOffer function

[SWS\_DM\_01368]{DRAFT} Definition of API function ara::diag::DoIPEntityIdentification::StopOffer [

<b>Kind:</b>	function	
<b>Header file:</b>	#include "ara/diag/doip_entity_identification.h"	
<b>Scope:</b>	<a href="#">class ara::diag::DoIPEntityIdentification</a>	
<b>Symbol:</b>	StopOffer()	
<b>Syntax:</b>	void StopOffer ();	
<b>Return value:</b>	None	
<b>Exception Safety:</b>	not exception safe	
<b>Description:</b>	This StopOffer will disable the forwarding of request messages from DM.	

]([SRS\\_Eth\\_00026](#))

## **C Interfaces to other Functional Clusters (informative)**

### **C.1 Overview**

AUTOSAR decided not to standardize interfaces which are exclusively used between Functional Clusters (on platform-level only), to allow efficient implementations, which might depend e.g. on the used Operating System.

This chapter provides informative guidelines how the interaction between Functional Clusters looks like, by clustering the relevant requirements of this document to describe Inter-Functional Cluster (IFC) interfaces. In addition, the standardized public interfaces which are accessible by user space applications (see chapter 8) can also be used for interaction between Functional Clusters.

The goal is to provide a clear understanding of Functional Cluster boundaries and interaction, without specifying syntactical details. This ensures compatibility between documents specifying different Functional Clusters and supports parallel implementation of different Functional Clusters. Details of the interfaces are up to the platform provider. Additional interfaces, parameters and return values can be added.

### **C.2 Interface Tables**

## D 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.

### D.1 Constraint and Specification Item History of this document according to AUTOSAR Release 17-10

#### D.1.1 Added Specification Items in 17-10

Number	Heading
[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> during Response Transmission
[SWS_DM_00281]	Cancellation of <code>Active Protocol</code> in Non-Default Session
[SWS_DM_00282]	Handling of <code>CurrentActiveProtocols</code>
[SWS_DM_00284]	<code>SecurityAccess</code> 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 <code>TRUE</code>
[SWS_DM_00289]	Configurable environmental condition check evaluates to <code>FALSE</code>
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response
[SWS_DM_00291]	<code>UdsMessage</code> class
[SWS_DM_00292]	<code>UdsMessage</code> non public constructors
[SWS_DM_00293]	<code>UdsMessage</code> Address type
[SWS_DM_00294]	meta info map type
[SWS_DM_00295]	meta info map vendor type
[SWS_DM_00296]	<code>TargetAddressType</code> Address type
[SWS_DM_00297]	<code>GetSa</code> method
[SWS_DM_00298]	<code>GetTa</code> method
[SWS_DM_00299]	<code>GetTaType</code> method
[SWS_DM_00300]	<code>GetPayload</code> method readonly
[SWS_DM_00301]	<code>GetPayload</code> method
[SWS_DM_00302]	<code>AddMetaInfo</code> method
[SWS_DM_00303]	<code>UdsMessage</code> Pointer
[SWS_DM_00304]	Const <code>UdsMessage</code> Pointer





Number	Heading
[SWS_DM_00305]	Const UdsMessage Pointer vendor type
[SWS_DM_00306]	UdsTransportProtocolMgr class
[SWS_DM_00307]	TransmissionResult type
[SWS_DM_00308]	Global Channel Identifier type
[SWS_DM_00309]	IndicateMessage method
[SWS_DM_00310]	NotifyMessageFailure method
[SWS_DM_00311]	HandleMessage method
[SWS_DM_00312]	TransmitConfirmation method
[SWS_DM_00313]	ChannelReestablished method
[SWS_DM_00314]	HandlerStopped method
[SWS_DM_00315]	UdsTransportProtocolHandler class
[SWS_DM_00316]	Header file
[SWS_DM_00317]	UdsTransportProtocolHandler constructor
[SWS_DM_00318]	UdsTransportProtocolHandler destructor
[SWS_DM_00319]	Initialize method
[SWS_DM_00320]	UdsTransportProtocolHandler UdsTransportProtocolMgr member
[SWS_DM_00321]	constructor member initialization
[SWS_DM_00322]	Start method
[SWS_DM_00323]	Stop method
[SWS_DM_00324]	UdsTransportProtocolHandler UdsTransportProtocolHandlerID member
[SWS_DM_00325]	GetHandlerID method
[SWS_DM_00326]	NotifyReestablishment method
[SWS_DM_00327]	Transmit method
[SWS_DM_00328]	UdsMessage Pointer vendor type
[SWS_DM_00329]	Lifecycle management of an Uds Transport Protocol implementation
[SWS_DM_00330]	Construction of an Uds Transport Protocol implementation
[SWS_DM_00331]	Initialization of an Uds Transport Protocol implementation
[SWS_DM_00332]	Starting of an Uds Transport Protocol implementation
[SWS_DM_00333]	Stopping of an Uds Transport Protocol implementation
[SWS_DM_00334]	UdsTransportProtocolMgr may be an abstract class
[SWS_DM_00335]	Header file
[SWS_DM_00336]	UdsTransportProtocolHandlerID
[SWS_DM_00337]	ChannelID
[SWS_DM_00338]	ByteVector
[SWS_DM_00339]	ByteVector vendor type
[SWS_DM_00340]	Waiting for Stop confirmation
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00343]	Acceptance of UDS message reception





Number	Heading
[SWS_DM_00344]	Refusal of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00363]	Positive response processing
[SWS_DM_00364]	Negative response processing
[SWS_DM_00365]	Suppression of response
[SWS_DM_00366]	Suppression of response for functional requests
[SWS_DM_00367]	No service processing
[SWS_DM_00368]	Sending busy responses
[SWS_DM_00369]	Max. number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19

**Table D.1: Added Specification Items in 17-10**

### D.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 D.2: Changed Specification Items in 17-10**

### D.1.3 Deleted Specification Items in 17-10

Number	Heading
[SWS_DM_00001]	Availability of operation cycle service interfaces
[SWS_DM_00053]	Cancellation of Active Protocol
[SWS_DM_00054]	Generic UDS Service Interface





Number	Heading
[SWS_DM_00073]	Checking enable conditions after status reports
[SWS_DM_00075]	Fulfilled enable conditions
[SWS_DM_00076]	Checking storage conditions in case the storage of event-related data is triggered
[SWS_DM_00077]	Checking storage conditions in case the update of event-related data is triggered
[SWS_DM_00081]	Routine Service Interface
[SWS_DM_00093]	Service Validation Interface
[SWS_DM_00094]	Data Services Interface
[SWS_DM_00149]	DTC related data
[SWS_DM_00157]	Snapshot record record data layout
[SWS_DM_00171]	Check for Supported DataIdentifier
[SWS_DM_00187]	Check for Supported DataIdentifier
[SWS_DM_00204]	Reaction on Unsupported Subfunction
[SWS_DM_00251]	Check for Supported Subfunction
[SWS_DM_CON-STR_00275]	Response processing after the actual reset

**Table D.3: Deleted Specification Items in 17-10**

#### **D.1.4 Added Constraints in 17-10**

none

#### **D.1.5 Changed Constraints in 17-10**

none

#### **D.1.6 Deleted Constraints in 17-10**

none

## D.2 Constraint and Specification Item History of this document according to AUTOSAR Release 18-03

### D.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 <code>internal DiagnosticDataElements</code>
[SWS_DM_00397]	Retrieving data for <code>external DiagnosticDataElements</code>
[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 <code>DiagnosticDataIdentifier</code>
[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 <code>DiagnosticDataIdentifier</code>
[SWS_DM_00408]	Retrieving data for requested DataIdentifier
[SWS_DM_00409]	Check supported DataIdentifier
[SWS_DM_00410]	Check session permission
[SWS_DM_00411]	Check security level permission
[SWS_DM_00412]	Check requested number of DataIdentifiers





Number	Heading
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00415]	Check supported DataIdentifier
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00434]	Providing the PowerMode in DoIP protocol messages
[SWS_DM_CON-STR_00394]	Internal DiagnosticDataElements are read-only
[SWS_DM_CON-STR_00395]	Restriction on DEM-exclusive DiagnosticDataElements
[SWS_DM_CON-STR_00396]	Restriction on DCM-exclusive DiagnosticDataElements

**Table D.4: Added Specification Items in 18-03**

## D.2.2 Changed Specification Items in 18-03

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00005]	DoIP Support
[SWS_DM_00007]	Uniqueness of diagnostic events
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00013]	Events without debouncing
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00017]	Calculation of the FDC based on the internal debounce counter
[SWS_DM_00018]	Internal debounce counter init and storage
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00022]	Debounce counter jump up behavior
[SWS_DM_00023]	Debounce counter jump down behavior
[SWS_DM_00024]	Qualified failed event using counter-based debouncing





Number	Heading
[SWS_DM_00025]	Qualified passed event using counter-based debouncing
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00028]	Debounce counter persistency
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00033]	Debounce timer behavior upon reported failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00036]	Debounce timer behavior upon reported passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00039]	Resetting the debounce counter upon starting or restarting an operation cycle
[SWS_DM_00040]	Definition of debounce counter reset
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00042]	Cancelling external service processors
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00046]	Each Diagnostic Protocol has own session resources
[SWS_DM_00047]	Each Diagnostic Protocol has own security-level resources
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00049]	Refusal of second diagnostic request from different diagnostic client with BusyRepeatRequest
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00055]	Supported event memories
[SWS_DM_00057]	Availability of a user-defined event memory
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00060]	Set of supported DTCs
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[13] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord
[SWS_DM_00064]	Definition of DTC groups
[SWS_DM_00065]	Always supported availability of the group of all DTCs
[SWS_DM_00069]	Monitor initialization for enable condition reenabling reason





Number	Heading
[SWS_DM_00070]	Monitor initialization for <a href="#">DTC</a> setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[ <a href="#">SWS_DM_00085</a> ]	Internal debounce counter init
[ <a href="#">SWS_DM_00086</a> ]	Resetting the debounce counter after clearing DTC
[SWS_DM_00087]	Enable condition influence on debouncing behavior (freeze)
[ <a href="#">SWS_DM_00088</a> ]	ControlDTCSetting influence (freeze)
[SWS_DM_00089]	Reporting PREPASSED or PREFAILED for events without assigned debouncing algorithm
[ <a href="#">SWS_DM_00090</a> ]	Support of UDS service ClearDiagnosticInformation
[ <a href="#">SWS_DM_00091</a> ]	Evaluation of ClearDiagnosticInformation parameters
[ <a href="#">SWS_DM_00092</a> ]	Parameter range check for groupOfDTC request parameter
[ <a href="#">SWS_DM_00096</a> ]	Validation Steps and Order
[ <a href="#">SWS_DM_00097</a> ]	Abort on failed verification step
[ <a href="#">SWS_DM_00111</a> ]	Configurable environment condition checks
[ <a href="#">SWS_DM_00112</a> ]	Condition check definition
[ <a href="#">SWS_DM_00113</a> ]	Positive response for UDS service 0x14
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[ <a href="#">SWS_DM_00115</a> ]	Memory error handling while clearing DTCs
[ <a href="#">SWS_DM_00116</a> ]	Clearing a <a href="#">DTC group</a>
[ <a href="#">SWS_DM_00117</a> ]	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
[ <a href="#">SWS_DM_00121</a> ]	Forbidden clearing of snapshot records and extended data records
[ <a href="#">SWS_DM_00122</a> ]	UDS response behavior on not allowed clear operations
[ <a href="#">SWS_DM_00123</a> ]	Block status byte clearing during a clear DTC operation
[ <a href="#">SWS_DM_00124</a> ]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[ <a href="#">SWS_DM_00128</a> ]	Realisation of UDS service 0x34 RequestDownload
[ <a href="#">SWS_DM_00129</a> ]	Supported addressAndLengthFormatIdentifier
[ <a href="#">SWS_DM_00130</a> ]	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
[ <a href="#">SWS_DM_00144</a> ]	Parallel clearing DTCs in different <a href="#">DiagnosticMemoryDestination</a>





Number	Heading
[SWS_DM_00145]	Allow only one simultaneous clear DTC operation for one <a href="#">DiagnosticMemoryDestination</a>
[SWS_DM_00146]	Unlock clear DTC operation for one <a href="#">DiagnosticMemoryDestination</a>
[SWS_DM_00147]	Behavior while trying to clear DTCs on a locked <a href="#">DiagnosticMemoryDestination</a>
[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 <a href="#">GroupOfAllDTCs</a>
[SWS_DM_00160]	Allow to clear single <a href="#">DTCs</a>
[SWS_DM_00161]	Negative response on not supported <a href="#">GroupOfDTC</a> parameter
[SWS_DM_00162]	Point in time for positive response for <a href="#">ClearDTC</a>
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of <a href="#">DiagnosticMonitor</a> service interfaces
[SWS_DM_00177]	Reaction on <a href="#">ApplicationError</a>
[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 <a href="#">ClearDiagnosticInformation</a>
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00205]	Providing the <a href="#">VIN</a> 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 <a href="#">ClearDiagnosticInformation</a> 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 <a href="#">ControlDTCSetting</a>







Number	Heading
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00238]	Aging and healing
[SWS_DM_00239]	Aging counter
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Reoccurrence after aging
[SWS_DM_00243]	Aging-related UDS status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00258]	Cancellation of <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





Number	Heading
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response
[SWS_DM_00309]	IndicateMessage method
[SWS_DM_00316]	Header file
[SWS_DM_00329]	Lifecycle management of an Uds Transport Protocol implementation
[SWS_DM_00330]	Construction of an Uds Transport Protocol implementation
[SWS_DM_00331]	Initialization of an Uds Transport Protocol implementation
[SWS_DM_00332]	Starting of an Uds Transport Protocol implementation
[SWS_DM_00333]	Stopping of an Uds Transport Protocol implementation
[SWS_DM_00335]	Header file
[SWS_DM_00340]	Waiting for Stop confirmation
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00363]	Unsupported Subfunction
[SWS_DM_00366]	Suppression of response for functional requests
[SWS_DM_00369]	Max. number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_CON-STR_00059]	Restriction on supported DTC format
[SWS_DM_CON-STR_00082]	Restriction on the configuration of the DTC group GroupOfAllDTCs
[SWS_DM_CON-STR_00084]	Each DTC shall be assigned to an event memory destination





Number	Heading
[SWS_DM_CONSTR_00168]	Required operation cycles for diagnostic events
[SWS_DM_CONSTR_00206]	Supported format for data identifier for VINDataIdentifier
[SWS_DM_CONSTR_00207]	Required VINDataIdentifier

**Table D.5: Changed Specification Items in 18-03**

### D.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 D.6: Deleted Specification Items in 18-03**

### D.2.4 Added Constraints in 18-03

none

### D.2.5 Changed Constraints in 18-03

none

### D.2.6 Deleted Constraints in 18-03

none

## D.3 Constraint and Specification Item History of this document according to AUTOSAR Release 18-10

### D.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 <a href="#">AAs</a>
[SWS_DM_00436]	Providing the <a href="#">GID</a> in DoIP protocol messages
[SWS_DM_00437]	Check supported RoutineIdentifier on active security level
[SWS_DM_00438]	Check Support of UDS service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of UDS service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of UDS service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of UDS service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of UDS service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of UDS service RequestTransferExit (0x37) on active security level
[SWS_DM_00444]	Check Support of UDS service ControlDTCSetting (0x85) in active session





Number	Heading
[SWS_DM_00445]	Check Support of UDS service ControlDTCSetting (0x85) on active security level
[SWS_DM_00446]	Check Support of UDS service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of UDS service RequestDownload (0x34) on active security level
[SWS_DM_00448]	Check supported RoutineIdentifier in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00475]	DoIP Version
[SWS_DM_00476]	User Controlled Warning IndicatorRequest-bit
[SWS_DM_00477]	Not Storing of 'warningIndicatorRequested' bit
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00480]	Security Access Blocking Timer
[SWS_DM_00481]	Handling of <a href="#">DiagnosticClearConditions</a>
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00483]	Cancellation trigger from <a href="#">AAs</a>
[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 resources
[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





Number	Heading
[SWS_DM_00508]	Reading DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00509]	Writing DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00651]	NumberOfStoredEntries
[SWS_DM_09010]	
[SWS_DM_09012]	
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_09017]	
[SWS_DM_09021]	
[SWS_DM_09028]	
[SWS_DM_CON-STR_00208]	Delay time value for sharedTimer
[SWS_DM_NA]	

**Table D.7: Added Specification Items in 18-10**

### D.3.2 Changed Specification Items in 18-10

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00004]	Operation cycle persistency
[SWS_DM_00005]	DoIP Support
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00011]	Selectability of parallelism mode
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00016]	Configurable number of supported parallel Diagnostic Conversations
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00046]	Each Diagnostic Conversation has its own session resources
[SWS_DM_00047]	Each Diagnostic Conversation has its own security-level resources
[SWS_DM_00049]	Refusal of diagnostic request due to prioritization with BusyRepeatRequest





Number	Heading
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[13] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00089]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00091]	Evaluation of ClearDiagnosticInformation parameters
[SWS_DM_00092]	Parameter range check for groupOfDTC request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00098]	UDS message checks
[SWS_DM_00099]	Supported Service SID level checks
[SWS_DM_00100]	Supported Service subfunction level checks
[SWS_DM_00101]	Session Access SID level Permission
[SWS_DM_00102]	Session Access subfunction level Permission
[SWS_DM_00103]	Security Access level Permission
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service 0x14
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block status byte clearing during a clear DTC operation
[SWS_DM_00124]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00126]	Realisation of UDS service 0x3E TesterPresent
[SWS_DM_00127]	Availability of diagnostic service processors





Number	Heading
[SWS_DM_00128]	Realization of UDS service RequestDownload (0x34)
[SWS_DM_00129]	Supported addressAndLengthFormatIdentifier
[SWS_DM_00130]	Not supported addressAndLengthFormatIdentifier
[SWS_DM_00131]	UDS service RequestDownload (0x34) processing
[SWS_DM_00134]	Realization of UDS service RequestUpload (0x35)
[SWS_DM_00136]	UDS service RequestUpload (0x35) processing
[SWS_DM_00137]	Realization of UDS service TransferData (0x36)
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00140]	Realisation of UDS service 0x28 CommunicationControl
[SWS_DM_00141]	Realization of UDS service RequestTransferExit (0x37)
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00159]	Allow only to clear <a href="#">GroupOfAllDTCs</a>
[SWS_DM_00160]	Allow to clear single <a href="#">DTCs</a>
[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 <a href="#">DiagnosticMonitor</a> service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00170]	Realisation of UDS service ReadDataByIdentifier (0x22)
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00186]	Realisation of UDS service WriteDataByIdentifier (0x2E)
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00193]	Support of a user-defined fault memory clear request
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnosticInformation
[SWS_DM_00195]	Clearing a user-defined memory
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00199]	Positive Response processing
[SWS_DM_00201]	Realization of UDS service RoutineControl (0x31)
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction







Number	Heading
[SWS_DM_00205]	Providing the <b>VIN</b> in DoIP protocol messages
[SWS_DM_00208]	Validation of the requested user-defined memory number
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00213]	DTC status processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the <b>DTC</b>
[SWS_DM_00216]	<b>DTC</b> status bit transitions triggered by operation cycle changes
[SWS_DM_00217]	<b>DTC</b> status bit transitions triggered by ClearDiagnosticInformation <b>UDS</b> service
[SWS_DM_00218]	Confirmation
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about DTC status changes
[SWS_DM_00222]	Observability of indicator status
[SWS_DM_00226]	Support of UDS service DiagnosticSessionControl
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00228]	Switch to requested Diagnostic Session
[SWS_DM_00229]	Support of UDS service ControlDTCSetting (0x85)
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Re-occurrence after aging
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00252]	Reaction on Unsupported Subfunction





Number	Heading
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00263]	ClearDTC call on invalid DTC or DTCgroup
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00266]	ClearDTC processing in case of memory errors
[SWS_DM_00267]	Possible failure of ClearDTC
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00273]	Notification event upon <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]	





Number	Heading
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	
[SWS_DM_00326]	
[SWS_DM_00327]	
[SWS_DM_00329]	Lifecycle management of an Uds Transport Protocol implementation
[SWS_DM_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00364]	Negative response processing
[SWS_DM_00365]	Suppression of positive response in accordance to ISO 14229-1[13]
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[13]
[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 <code>internal DiagnosticDataElements</code>
[SWS_DM_00397]	Retrieving data for <code>external DiagnosticDataElements</code>
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00404]	Default Service Interface for reading <code>DiagnosticDataIdentifier</code>





Number	Heading
[SWS_DM_00407]	Default Service Interface for writing <a href="#">DiagnosticDataIdentifier</a>
[SWS_DM_00408]	Retrieving data for requested DataIdentifier
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00434]	Providing the <a href="#">PowerMode</a> in DoIP protocol messages

**Table D.8: Changed Specification Items in 18-10**

### D.3.3 Deleted Specification Items in 18-10

Number	Heading
[SWS_DM_00001]	SRS Diagnostics
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00051]	Cancellation of Active Protocol with lower priority
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00066]	Monitor initialization
[SWS_DM_00105]	Configurable Manufacturer Permission Check Services
[SWS_DM_00107]	Configurable Supplier Permission Check Services
[SWS_DM_00118]	Event specific configuration to allow clearing of a DTC
[SWS_DM_00119]	Init value for events with clear allowed information
[SWS_DM_00120]	Description of application interface to control the clear event behavior
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[SWS_DM_00161]	Negative response on not supported GroupOfDTC parameter
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment





Number	Heading
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00185]	No Match
[SWS_DM_00258]	Cancellation of Active Protocol in non-default session
[SWS_DM_00259]	Completion of already Active Protocols in default session
[SWS_DM_00274]	Definition of an active Diagnostic Protocol
[SWS_DM_00281]	Cancellation of active DiagnosticConversation in Non-Default Session
[SWS_DM_00282]	Handling of non-/active diagnostic conversations
[SWS_DM_00295]	meta info map vendor type
[SWS_DM_00305]	Const UdsMessage Pointer vendor type
[SWS_DM_00308]	Global Channel Identifier type
[SWS_DM_00316]	Header file
[SWS_DM_00317]	UdsTransportProtocolHandler constructor
[SWS_DM_00318]	UdsTransportProtocolHandler destructor
[SWS_DM_00320]	UdsTransportProtocolHandler UdsTransportProtocolMgr member
[SWS_DM_00321]	constructor member initialization
[SWS_DM_00324]	UdsTransportProtocolHandler UdsTransportProtocolHandlerID member
[SWS_DM_00328]	UdsMessage Pointer vendor type
[SWS_DM_00334]	UdsTransportProtocolMgr may be an abstract class
[SWS_DM_00335]	Header file
[SWS_DM_00339]	ByteVector vendor type
[SWS_DM_00402]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00403]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00405]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00406]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00410]	Check session permission
[SWS_DM_00411]	Check security level permission
[SWS_DM_CON-STR_00207]	Required VINDataIdentifier

**Table D.9: Deleted Specification Items in 18-10**

### D.3.4 Added Constraints in 18-10

none

### D.3.5 Changed Constraints in 18-10

none

### D.3.6 Deleted Constraints in 18-10

none

## D.4 Constraint and Specification Item History of this document according to AUTOSAR Release 19-03

### D.4.1 Added Specification Items in 19-03

Number	Heading
[SWS_DM_00510]	Namespace of Service header files
[SWS_DM_00511]	Implementation Types header files existence
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00513]	Implementation Types header file namespace
[SWS_DM_00526]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00561]	Deployment of diagnostic PortInterfaces





Number	Heading
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00566]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00569]	Handling of storage conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00600]	





Number	Heading
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00617]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00620]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00644]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	







Number	Heading
[SWS_DM_00658]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00691]	
[SWS_DM_00692]	
[SWS_DM_00693]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00710]	
[SWS_DM_00711]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00731]	
[SWS_DM_00732]	



△

Number	Heading
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00750]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00754]	
[SWS_DM_00755]	
[SWS_DM_00756]	
[SWS_DM_00781]	NumberOfStoredEntries
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00784]	
[SWS_DM_00785]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00793]	
[SWS_DM_00794]	
[SWS_DM_00795]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_00803]	

**Table D.10: Added Specification Items in 19-03**

#### D.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[13]
[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





Number	Heading
[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_CON-STR_00395]	Restriction on DEM-exclusive <a href="#">DiagnosticDataElements</a>

**Table D.11: Changed Specification Items in 19-03**

#### D.4.3 Deleted Specification Items in 19-03

Number	Heading
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00483]	Cancellation trigger from <a href="#">AAs</a>
[SWS_DM_09028]	

**Table D.12: Deleted Specification Items in 19-03**

#### D.4.4 Added Constraints in 19-03

none

#### D.4.5 Changed Constraints in 19-03

none

#### D.4.6 Deleted Constraints in 19-03

none

## D.5 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11

### D.5.1 Added Specification Items in 19-11

Number	Heading
[SWS_DM_00450]	Security Access subfunction level Permission
[SWS_DM_00502]	Support for Custom Diagnostic Services
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00645]	
[SWS_DM_00659]	
[SWS_DM_00660]	
[SWS_DM_00661]	
[SWS_DM_00662]	
[SWS_DM_00690]	
[SWS_DM_00702]	
[SWS_DM_00730]	
[SWS_DM_00760]	
[SWS_DM_00761]	
[SWS_DM_00762]	
[SWS_DM_00763]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00767]	
[SWS_DM_00770]	
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00804]	
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	





Number	Heading
[SWS_DM_00809]	
[SWS_DM_00810]	
[SWS_DM_00811]	Re-enabling of ControlDTCSetting by Diagnostic Application
[SWS_DM_00812]	Re-enabling on transition to default session
[SWS_DM_00813]	Providing the <code>GID</code> in DoIP protocol messages
[SWS_DM_00814]	Providing the <code>PowerMode</code> 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 <code>AA</code> s
[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 <code>DataIdentifier</code> interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by <code>GenericUDSService</code> interface
[SWS_DM_00850]	Default Service Interface for reading <code>DiagnosticDataIdentifier</code>
[SWS_DM_00855]	Providing the <code>VIN</code> in DoIP protocol messages
[SWS_DM_00856]	Initial values for Diagnostic Conversation
[SWS_DM_00857]	Signature of Manufacturer Permission Check Method
[SWS_DM_00858]	Signature of Supplier Permission Check Method
[SWS_DM_00859]	Confirmation of service processing
[SWS_DM_00860]	No service processing
[SWS_DM_00861]	Negative response processing





Number	Heading
[SWS_DM_00862]	Suppression of negative response for functional requests in accordance to ISO 14229-1[13]
[SWS_DM_00863]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00864]	Checking Supported Subfunction for CompareKey
[SWS_DM_00865]	Communication control service processing
[SWS_DM_00866]	Negative Response processing
[SWS_DM_00867]	UDS service RequestDownload (0x34) processing
[SWS_DM_00868]	UDS service RequestUpload (0x35) processing
[SWS_DM_00869]	UDS service TransferData (0x36) processing
[SWS_DM_00870]	UDS service TransferData (0x36) validation
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00872]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00873]	Diagnostic event processing interface
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00877]	Starting time-based event debouncing for failed
[SWS_DM_00878]	Starting time-based event debouncing for passed
[SWS_DM_00879]	Application resetting the debounce counter
[SWS_DM_00880]	Debounce time freeze request
[SWS_DM_00881]	Enable condition influence on debouncing behavior (freeze)
[SWS_DM_00882]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00883]	UDS DTC status bit transitions triggered by test results
[SWS_DM_00884]	Resetting the status of the DTC
[SWS_DM_00885]	UDS DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00886]	Observability of the status byte
[SWS_DM_00887]	Notification about DTC status changes
[SWS_DM_00888]	Observability of indicator status
[SWS_DM_00889]	Automatic starting of operation cycles
[SWS_DM_00890]	Automatic ending of operation cycles
[SWS_DM_00891]	Restart of operation cycles
[SWS_DM_00892]	Operation cycles are only ended once
[SWS_DM_00893]	Triggering for snapshot record storage
[SWS_DM_00894]	Notification event upon snapshot record updates
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00896]	Handling of DiagnosticClearConditions
[SWS_DM_00897]	Usage of ClearDTC Interface





Number	Heading
[SWS_DM_00898]	ClearDTC call on invalid <code>DTC</code> 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]	NumberOfStoredEntries
[SWS_DM_00903]	Reading DiagnosticDataIdentifier configured for representing <code>VIN</code>
[SWS_DM_00904]	Writing DiagnosticDataIdentifier configured for representing <code>VIN</code>
[SWS_DM_00905]	Retrieving data for <code>external DiagnosticDataElements</code>
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00907]	Default Service Interface for writing <code>DiagnosticDataIdentifier</code>
[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 D.13: Added Specification Items in 19-11**

## D.5.2 Changed Specification Items in 19-11

Number	Heading
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00024]	Qualified failed event using counter-based debouncing
[SWS_DM_00025]	Qualified passed event using counter-based debouncing
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00033]	Debounce timer behavior upon reported failed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00036]	Debounce timer behavior upon reported passed
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00217]	<code>UDS DTC status bit</code> transitions triggered by <code>ClearDiagnosticInformation</code> <code>UDS service</code>
[SWS_DM_00218]	Trip Counter
[SWS_DM_00242]	Re-occurrence after <code>Aging</code>







Number	Heading
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00296]	
[SWS_DM_00307]	
[SWS_DM_00393]	Retrieving data for internal DiagnosticDataElements
[SWS_DM_00401]	Reading Diagnostic Data Identifier on Data Element level
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00421]	Identification of a Diagnostic Client
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00426]	Assigning a UDS request to an existing Diagnostic Conversation
[SWS_DM_00427]	Priority of a Diagnostic Conversation
[SWS_DM_00428]	Treatment of priority values
[SWS_DM_00429]	Prioritization in active non-default session
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[SWS_DM_00433]	Refusal of diagnostic request due to busy Diagnostic Conversation
[SWS_DM_00437]	Check supported RoutineIdentifier subfunction on active security level
[SWS_DM_00448]	Check supported RoutineIdentifier subfunction in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00475]	DoIP Version
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00507]	Length check on UDS Service 0x27 request with Subfunction for Request-Seed
[SWS_DM_00526]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	





Number	Heading
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for <b>DTC</b> setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of <b>enable conditions</b>
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	





Number	Heading
[SWS_DM_00616]	
[SWS_DM_00618]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00640]	
[SWS_DM_00644]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	
[SWS_DM_00652]	
[SWS_DM_00653]	
[SWS_DM_00654]	
[SWS_DM_00655]	
[SWS_DM_00656]	
[SWS_DM_00657]	
[SWS_DM_00658]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00692]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	





Number	Heading
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00712]	
[SWS_DM_00713]	
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00731]	
[SWS_DM_00732]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00750]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00754]	
[SWS_DM_00755]	
[SWS_DM_00756]	
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	





Number	Heading
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_09012]	
[SWS_DM_09017]	

**Table D.14: Changed Specification Items in 19-11**

### D.5.3 Deleted Specification Items in 19-11

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00011]	Selectability of parallelism mode
[SWS_DM_00016]	Configurable number of supported parallel Diagnostic Conversations
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00042]	Canceling external service processors
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00087]	Enable condition influence on debouncing behavior (freeze)





Number	Heading
[SWS_DM_00089]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00131]	UDS service RequestDownload (0x34) processing
[SWS_DM_00136]	UDS service RequestUpload (0x35) processing
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of DiagnosticMonitor service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00205]	Providing the VIN in DoIP protocol messages
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00216]	DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about DTC status changes
[SWS_DM_00222]	Observability of indicator status
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00263]	ClearDTC call on invalid DTC or DTCgroup
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00266]	ClearDTC processing in case of memory errors



△

Number	Heading
[SWS_DM_00267]	Possible failure of ClearDTC
[SWS_DM_00273]	Notification event upon <code>snapshot record</code> 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[13]
[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 <code>external DiagnosticDataElements</code>
[SWS_DM_00404]	Default Service Interface for reading <code>DiagnosticDataIdentifier</code>
[SWS_DM_00407]	Default Service Interface for writing <code>DiagnosticDataIdentifier</code>
[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 <code>PowerMode</code> in DoIP protocol messages
[SWS_DM_00435]	Default session change trigger from <code>AAs</code>
[SWS_DM_00436]	Providing the <code>GID</code> 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 <code>DiagnosticClearConditions</code>
[SWS_DM_00484]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00485]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00503]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00504]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00508]	Reading DiagnosticDataIdentifier configured for representing <code>VIN</code>
[SWS_DM_00509]	Writing DiagnosticDataIdentifier configured for representing <code>VIN</code>
[SWS_DM_00566]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00569]	Handling of storage conditions
[SWS_DM_00781]	NumberOfStoredEntries

**Table D.15: Deleted Specification Items in 19-11**

**D.5.4 Added Constraints in 19-11**

none

**D.5.5 Changed Constraints in 19-11**

none

**D.5.6 Deleted Constraints in 19-11**

none

**D.6 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11**

**D.6.1 Added Specification Items in R20-11**

Number	Heading
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	
[SWS_DM_00521]	
[SWS_DM_00522]	
[SWS_DM_00523]	
[SWS_DM_00524]	
[SWS_DM_00525]	
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	







Number	Heading
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	
[SWS_DM_00786]	
[SWS_DM_00916]	Priority values
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00920]	Configuration of the event memory size
[SWS_DM_00921]	Configuration of Error Memory Overflow Indication as extended data record
[SWS_DM_00922]	Persistent storage for event memory overflow information
[SWS_DM_00923]	Event memory overflow set condition
[SWS_DM_00924]	Event memory overflow reset condition
[SWS_DM_00925]	Event memory overflow notifier on 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]	





Number	Heading
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for <b>DID</b> 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 <b>FDC</b> " as extended data record
[SWS_DM_00952]	Configuration of DTC "max. <b>FDC</b> since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. <b>FDC</b> current cycle" as extended data record
[SWS_DM_00954]	Configuration of DTC "occurrence counter" as extended data record
[SWS_DM_00955]	Configuration of DTC "aging counter up/down" as extended data record
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00957]	Configuration of DTC "aging counter down" as extended data record
[SWS_DM_00958]	Default value for DTC "aging counter up" if aging is not allowed
[SWS_DM_00959]	Default value for DTC "aging counter down" if aging is not allowed
[SWS_DM_00961]	Configuration of a DTCs significance as extended data record
[SWS_DM_00962]	Configuration of a DTCs Failed Operation Cycles as extended data record
[SWS_DM_00963]	Configuration of a DTCs failed operation Cycles Since First Failed as extended data record
[SWS_DM_00964]	Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record
[SWS_DM_00965]	Caching of monitor results
[SWS_DM_00966]	Reporting of DTCStatusAvailabilityMask
[SWS_DM_00967]	Support of <b>UDS</b> service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00970]	Behavior of failed data element retrieval
[SWS_DM_00971]	
[SWS_DM_00972]	
[SWS_DM_00973]	





Number	Heading
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00981]	Conditions of status based reporting order
[SWS_DM_00982]	Reporting order direction
[SWS_DM_00983]	Processing of Custom Diagnostic Services
[SWS_DM_00984]	Return of cancellation status
[SWS_DM_00989]	
[SWS_DM_00990]	
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01008]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01015]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01018]	ECUReset <a href="#">ara::diag::ResetRequestType</a> check





Number	Heading
[SWS_DM_01019]	Custom <code>ara::diag::ResetRequestType</code> processing
[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]	Positive response before reset assurance
[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> changes
[SWS_DM_01028]	<code>Event status</code> bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_01029]	Notification about <code>Event status</code> changes
[SWS_DM_01030]	Observability of the UDS <code>DTC status byte</code>
[SWS_DM_01031]	Notification about UDS <code>DTC</code> 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 <code>DiagnosticEvent.confirmationThreshold</code>
[SWS_DM_01038]	Reading Diagnostic Data Identifier by <code>ara::diag::GenericDataIdentifier</code> interface
[SWS_DM_01039]	Writing Diagnostic Data Identifier by <code>DataIdentifier</code> interface
[SWS_DM_01040]	Realization of UDS service <code>ReadDataByPeriodicIdentifier(0x2A)</code>
[SWS_DM_01041]	Check requested number of periodic <code>DataIdentifiers</code>
[SWS_DM_01042]	Minimum length check for <code>ReadDataByPeriodicIdentifier</code>
[SWS_DM_01043]	Check supported periodic <code>DataIdentifier</code>
[SWS_DM_01044]	Check Transmission Mode
[SWS_DM_01045]	Check Scheduler Availability
[SWS_DM_01046]	Check supported <code>DataIdentifier</code> in active session
[SWS_DM_01047]	Check supported <code>DataIdentifier</code> on active security level
[SWS_DM_01048]	Check <code>DataIdentifier</code> for environmental conditions
[SWS_DM_01049]	Checks Dynamically Defined <code>DIDs</code> in <code>ReadDataByPeriodicIdentifier</code>
[SWS_DM_01050]	Periodic <code>DID</code> length check
[SWS_DM_01051]	DM behavior on transmission Mode <code>stopSending</code> without periodic <code>DataIdentifier</code> in the request
[SWS_DM_01052]	DM behavior on transmission Mode <code>stopSending</code> with supported periodic <code>DataIdentifier</code> in the request
[SWS_DM_01053]	DM behavior on transmission Mode <code>stopSending</code> with not supported periodic <code>DataIdentifier</code> in the request





Number	Heading
[SWS_DM_01054]	Starting to transmit PDIDs after positive response
[SWS_DM_01055]	Reaction on ApplicationError
[SWS_DM_01056]	Optional condition checks for sending periodic DIDs
[SWS_DM_01057]	Optional stopping PDIDs after session change
[SWS_DM_01058]	Optional stopping PDIDs after security level change
[SWS_DM_01059]	No periodic DIDs in default session
[SWS_DM_01060]	Support of Scheduler type 1
[SWS_DM_01061]	Trigger all scheduled PDIDs per scheduler
[SWS_DM_01062]	Transmission of all PDIDs on the periodic connection
[SWS_DM_01063]	Transmission error behavior
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_01070]	Support of UDS service 0x2C in Adaptive AUTOSAR DM
[SWS_DM_01071]	No persistency of defined DDIDs
[SWS_DM_01072]	Persistency of defined DDIDs
[SWS_DM_01073]	DM behavior for subfunction 'defineByIdentifier'
[SWS_DM_01074]	Only static DIDs as sourceDataIdentifier
[SWS_DM_01075]	Maximum number of sourceDataIdentifiers in the request
[SWS_DM_01076]	Clearing all configured DDIDs
[SWS_DM_01077]	Clearing individual configured DDIDs
[SWS_DM_01078]	Clear DDIDs on session change
[SWS_DM_01079]	Session check for DDID
[SWS_DM_01080]	Security level check for DDID
[SWS_DM_01081]	Session check for sourceDataIdentifier
[SWS_DM_01082]	Security level check for sourceDataIdentifier
[SWS_DM_01083]	Use of configured DID ports to get DDID data
[SWS_DM_CON-STR_00960]	No support for DEM_AGINGCTR_UPCNT_FIRST_ACTIVE

**Table D.16: Added Specification Items in R20-11**

## D.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[13] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00123]	Block clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00124]	Limited clearing of UDS DTC status byte during a clear DTC operation
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00150]	Primary trigger for event memory entry storage
[SWS_DM_00213]	DTC status processing
[SWS_DM_00218]	UDS DTC status bit 'kConfirmedDTC'
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00237]	Aging
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00369]	Maximum number of busy responses
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00413]	Check supported DataIdentifier in active session
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00437]	Security Level check for RoutineIdentifier
[SWS_DM_00448]	Check supported RoutineIdentifier subfunction in active session
[SWS_DM_00502]	Support for Custom Diagnostic Services
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	





Number	Heading
[SWS_DM_00557]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of <i>enable conditions</i>
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00585]	
[SWS_DM_00587]	
[SWS_DM_00589]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00618]	
[SWS_DM_00619]	





Number	Heading
[SWS_DM_00634]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00640]	
[SWS_DM_00650]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00670]	
[SWS_DM_00673]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00699]	
[SWS_DM_00722]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00732]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00762]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00774]	
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00787]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00797]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_00806]	
[SWS_DM_00808]	
[SWS_DM_00809]	







Number	Heading
[SWS_DM_00836]	
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00857]	Signature of Manufacturer Permission Check Method
[SWS_DM_00858]	Signature of Supplier Permission Check Method
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00883]	UDS <code>DTC status bit</code> transitions triggered by test results
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00898]	ClearDTC call on invalid <code>DTC</code> or <code>DTC group</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_09015]	
[SWS_DM_09016]	
[SWS_DM_CON-STR_00059]	Restriction on supported <code>DTC</code> format
[SWS_DM_CON-STR_00396]	Restriction on DCM-exclusive <code>DiagnosticDataElements</code>

**Table D.17: Changed Specification Items in R20-11**

### D.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 <code>DTC</code>
[SWS_DM_00885]	UDS <code>DTC status bit</code> transitions triggered by <code>operation cycle</code> changes
[SWS_DM_00887]	Notification about <code>DTC</code> status changes

**Table D.18: Deleted Specification Items in R20-11**

#### D.6.4 Added Constraints in R20-11

none

#### D.6.5 Changed Constraints in R20-11

none

#### D.6.6 Deleted Constraints in R20-11

none

### D.7 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11

#### D.7.1 Added Specification Items in R21-11

Number	Heading
[SWS_DM_01084]	Triggering for snapshot record storage (calculated)
[SWS_DM_01085]	Triggering for snapshot record storage (configured, without update)
[SWS_DM_01086]	Triggering for snapshot record storage (configured, with update)
[SWS_DM_01087]	Snapshot record layout
[SWS_DM_01088]	
[SWS_DM_01089]	
[SWS_DM_01090]	Calling ExecuteReset() if positive response shall be sent after reset
[SWS_DM_01091]	Maximum time frame to trigger ExecuteReset()
[SWS_DM_01092]	EnableRapidPowerShutdown Positive Response
[SWS_DM_01093]	Caching of conditions
[SWS_DM_01094]	Monitor initialization for enable condition re-enabling reason and ControlDTCSetting set to On
[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





Number	Heading
[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 <i>DTCs event</i> with event combination
[SWS_DM_01109]	<i>UDS DTC</i> status update for combined <i>DTCs</i>
[SWS_DM_01110]	Callbacks for combined <i>UDS DTC</i> 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 <i>DEXT</i> parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate
[SWS_DM_01119]	Support of <i>DEXT</i> parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents
[SWS_DM_01120]	Support of <i>DEXT</i> parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of <i>DEXT</i> parameter DiagnosticResponseOnEventClass.maxSupportedDIDLenght
[SWS_DM_01122]	Support of <i>DEXT</i> 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]	





Number	Heading
[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]	





Number	Heading
[SWS_DM_01177]	
[SWS_DM_01178]	
[SWS_DM_01179]	
[SWS_DM_01180]	
[SWS_DM_01181]	
[SWS_DM_01182]	
[SWS_DM_01183]	
[SWS_DM_01184]	
[SWS_DM_01185]	
[SWS_DM_01186]	
[SWS_DM_01187]	
[SWS_DM_01188]	
[SWS_DM_01189]	
[SWS_DM_01190]	
[SWS_DM_01191]	
[SWS_DM_01192]	
[SWS_DM_01193]	
[SWS_DM_01194]	
[SWS_DM_01195]	
[SWS_DM_01196]	
[SWS_DM_01197]	
[SWS_DM_01198]	
[SWS_DM_01199]	
[SWS_DM_01200]	
[SWS_DM_01201]	
[SWS_DM_01202]	Get ClientAuthentication Instance
[SWS_DM_01203]	GetAll ClientAuthentication Instance
[SWS_DM_01204]	Default Authentication Role
[SWS_DM_01205]	Default Authentication State
[SWS_DM_01206]	Set AuthenticationRole
[SWS_DM_01207]	Get Authentication State
[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





Number	Heading
[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
[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 verifyCertificateBidirectional
[SWS_DM_01239]	Unexpected verifyCertificateBidirectional from a different client
[SWS_DM_01240]	Processing the proofOfOwnership request
[SWS_DM_01241]	Handling Negative return values of <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 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 <code>ara::diag::Authentication::TransmitCertificate</code> method





Number	Heading
[SWS_DM_01250]	Handling unspecified negative return values of <code>ara::diag::Authentication::TransmitCertificate</code> method
[SWS_DM_01251]	Successful verification of <code>transmitCertificate</code>
[SWS_DM_CONSTR_00961]	Limits of priority values

**Table D.19: Added Specification Items in R21-11**

### D.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 <code>DTC</code>
[SWS_DM_00088]	<code>ControlDTCSetting</code> influence (freeze)
[SWS_DM_00163]	Definition of a inhibited clear operation on single <code>DTC</code>
[SWS_DM_00164]	Definition of a inhibited clear operation for a group of <code>DTCs</code>
[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]	





Number	Heading
[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 <a href="#">UDS</a> service 0x86 ResponseOnEvent
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	
[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]	







Number	Heading
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	
[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 <code>enable conditions</code>
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	





Number	Heading
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00600]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[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]	





Number	Heading
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00644]	
[SWS_DM_00645]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	
[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]	
[SWS_DM_00723]	





Number	Heading
[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]	
[SWS_DM_00783]	
[SWS_DM_00784]	





Number	Heading
[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]	
[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





Number	Heading
[SWS_DM_00869]	UDS service TransferData (0x36) download processing
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) download processing
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00879]	Application resetting the internal debounce counter
[SWS_DM_00882]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00902]	NumberOfStoredEntries
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00971]	
[SWS_DM_00972]	
[SWS_DM_00973]	
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[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]	



△

Number	Heading
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01023]	Calling ExecuteReset() if positive response shall be sent before reset
[SWS_DM_01027]	Event status bit transitions triggered by operation cycle restarting
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_09010]	
[SWS_DM_09011]	
[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 D.20: Changed Specification Items in R21-11**



### D.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 <a href="#">DTC</a>
[SWS_DM_00438]	Check Support of <a href="#">UDS</a> service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of <a href="#">UDS</a> service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of <a href="#">UDS</a> service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of <a href="#">UDS</a> service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of <a href="#">UDS</a> service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of <a href="#">UDS</a> service RequestTransferExit (0x37) on active security level
[SWS_DM_00446]	Check Support of <a href="#">UDS</a> service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of <a href="#">UDS</a> 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 <a href="#">DTC</a> 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]	<a href="#">UDS</a> service TransferData (0x36) validation
[SWS_DM_00872]	<a href="#">UDS</a> 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 <a href="#">snapshot record</a> storage
[SWS_DM_00903]	Reading DiagnosticDataIdentifier configured for representing <a href="#">VIN</a>
[SWS_DM_00904]	Writing DiagnosticDataIdentifier configured for representing <a href="#">VIN</a>





Number	Heading
[SWS_DM_01008]	
[SWS_DM_01015]	

**Table D.21: Deleted Specification Items in R21-11**

#### D.7.4 Added Constraints in R21-11

none

#### D.7.5 Changed Constraints in R21-11

none

#### D.7.6 Deleted Constraints in R21-11

none

### D.8 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11

#### D.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 <a href="#">ara::diag::Conversation::GetDiagnosticSecurityLevelShortName</a>
[SWS_DM_01260]	Validation of Invalid Security Level in <a href="#">ara::diag::Conversation::GetDiagnosticSecurityLevelShortName</a>





Number	Heading
[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 <code>DTC</code> number for events without <code>DTC</code>
[SWS_DM_01270]	UDS Service RoutineControl (0x31) startRoutine processing with typed interface
[SWS_DM_01271]	UDS Service RoutineControl (0x31) stopRoutine processing with typed interface
[SWS_DM_01272]	UDS Service RoutineControl (0x31) requestRoutineResults processing with typed interface
[SWS_DM_01273]	Namespace for typed UDS Service RoutineControl (0x31)
[SWS_DM_01274]	Namespace for typed DiagnosticDataIdentifier interface
[SWS_DM_01275]	Namespace for typed DiagnosticDataElements interface
[SWS_DM_01276]	Triggering for <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords = 1</code> )
[SWS_DM_01277]	Triggering for <code>snapshot record</code> storage (calculated, <code>maxNumberFreezeFrameRecords &gt; 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 <code>DTCs</code>
[SWS_DM_01297]	Behavior of <code>DTCInformation::SetNumberOfStoredEntriesNotifier</code> for suppressed <code>DTCs</code>
[SWS_DM_01298]	Behavior of <code>DTCInformation::SetDTCStatusChangedNotifier</code> for suppressed <code>DTCs</code>





Number	Heading
[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 <code>ClearDiagnosticInformation</code> for suppressed DTCs inside a group
[SWS_DM_01303]	Behavior of <code>Diagnostic Client</code> services <code>ClearDiagnosticInformation</code> for suppressed DTCs
[SWS_DM_01304]	Behavior of services <code>ReadDTCInformation</code> for <code>ExtendedData</code> and <code>SnapshotData</code> for suppressed DTCs
[SWS_DM_01305]	Behavior of services <code>ReadDTCInformation</code> with <code>DTC</code> mask record for suppressed DTCs
[SWS_DM_01306]	Functionality of <code>DTCInformation::GetDtcSuppression</code>
[SWS_DM_01307]	Precondition for suppression
[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]	





Number	Heading
[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 MetalInfo
[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





Number	Heading
[SWS_DM_01370]	DNS-Based Service Discovery and Multicast DNS for SOVD
[SWS_DM_01371]	Secure Communication for SOVD using TLS
[SWS_DM_01372]	Representation of DM by SOVD component
[SWS_DM_01373]	Representation of Diagnostic Server instance by SOVD subcomponents
[SWS_DM_01374]	Dispatching of SOVD requests/responses
[SWS_DM_01375]	Behavior on locked SOVD
[SWS_DM_01376]	Response behavior of SOVD services without access rights
[SWS_DM_01379]	SOVD mode control_dtc_setting set value
[SWS_DM_01380]	SOVD mode control_dtc_setting get value
[SWS_DM_01381]	SOVD mode control_dtc_setting value schema
[SWS_DM_01382]	SOVD mode control_dtc_setting name
[SWS_DM_01383]	SOVD mode control_dtc_setting
[SWS_DM_01384]	SOVD mode communication_control set value
[SWS_DM_01385]	SOVD mode communication_control get value
[SWS_DM_01386]	SOVD mode communication_control value schema
[SWS_DM_01387]	SOVD mode communication_control name
[SWS_DM_01388]	SOVD mode communication_control
[SWS_DM_01389]	SOVD method Retrieve List of All Supported Modes of an Entity
[SWS_DM_01390]	SOVD operations request and response parameters
[SWS_DM_01391]	SOVD method Stop the Execution of an Operation proximity_response
[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





Number	Heading
[SWS_DM_01407]	SOVD method Delete All Faults of an Entity without scope
[SWS_DM_01408]	SOVD method Delete All Faults of an Entity with scope
[SWS_DM_01409]	SOVD method Delete All Faults of an Entity
[SWS_DM_01411]	SOVD fault environment_data query
[SWS_DM_01412]	SOVD fault environment_data
[SWS_DM_01413]	SOVD fault attribute symptom
[SWS_DM_01414]	SOVD fault attribute status
[SWS_DM_01415]	SOVD fault general attributes
[SWS_DM_01416]	SOVD method Read Details for a Fault
[SWS_DM_01417]	SOVD method Read Faults from an Entity
[SWS_DM_01418]	SOVD faults
[SWS_DM_01419]	SOVD method Read Multiple Data Values at Once from an Entity Using a Data List
[SWS_DM_01420]	SOVD data-list
[SWS_DM_01421]	SOVD method Write Configuration as Parameters data processing
[SWS_DM_01422]	SOVD method Write Configuration as Parameters
[SWS_DM_01423]	SOVD method Read Configuration as Parameters data query
[SWS_DM_01424]	SOVD method Read Configuration as Parameters
[SWS_DM_01425]	SOVD configuration attribute version and content_type
[SWS_DM_01426]	SOVD configuration attribute type
[SWS_DM_01427]	SOVD configuration attribute name
[SWS_DM_01428]	SOVD configuration attribute id
[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





Number	Heading
[SWS_DM_01444]	Data-groups <b>type</b> content
[SWS_DM_01445]	Data-groups <b>type</b>
[SWS_DM_01446]	<b>SOVD method</b> Retrieve Groups Supported by a Data Resource Collection
[SWS_DM_01447]	Data categories <b>type</b>
[SWS_DM_01448]	<b>Standard resource</b> Data categories
[SWS_DM_01449]	<b>SOVD method</b> Retrieve Categories Supported by a Data Resource Collection
[SWS_DM_01450]	SOVDInfo <b>type</b> content
[SWS_DM_01451]	SOVDInfo <b>type</b>
[SWS_DM_01452]	Mismatching versions
[SWS_DM_01453]	path to version-info
[SWS_DM_01454]	Query to the <b>SOVD API</b> version
[SWS_DM_01455]	<b>SOVD method</b> for <b>SOVD API</b> Versioning
[SWS_DM_01456]	<b>SOVD method</b> Query an Online Capability Description
[SWS_DM_01457]	<b>SOVD data</b> representation of arrays
[SWS_DM_01458]	<b>SOVD data</b> representation of units
[SWS_DM_01459]	<b>SOVD data</b> representation of textual Strings
[SWS_DM_01460]	<b>SOVD data</b> representation of <b>baseTypeEncoding</b> IEEE754
[SWS_DM_01461]	<b>SOVD data</b> representation of BITFIELD_TEXTTABLE
[SWS_DM_01462]	<b>SOVD data</b> representation of TAB_NOINTP
[SWS_DM_01463]	<b>SOVD data</b> representation of atomic scaled numeric data with texttable
[SWS_DM_01464]	<b>SOVD data</b> representation of TEXTTABLE
[SWS_DM_01465]	<b>SOVD data</b> representation of atomic numeric data
[SWS_DM_01466]	Environmental Condition Check for <b>SOVD</b> evaluated to FALSE
[SWS_DM_01467]	Environmental Condition Check for <b>SOVD</b> evaluated to TRUE
[SWS_DM_01468]	Check of Environmental Conditions before executing <b>SOVD</b> 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 <b>SOVD</b> entity after time expiration
[SWS_DM_01474]	DM shall allow access only to unlocked <b>SOVD</b> entities
[SWS_DM_01475]	DM shall allow only one lock per <b>SOVD</b> entity
[SWS_DM_01476]	Parallel <b>SOVD</b> client handling
[SWS_DM_01477]	<b>SOVD lock</b> in UDS extended session
[SWS_DM_01478]	
[SWS_DM_01479]	
[SWS_DM_01480]	







Number	Heading
[SWS_DM_01481]	
[SWS_DM_01482]	
[SWS_DM_01483]	
[SWS_DM_01484]	
[SWS_DM_01485]	
[SWS_DM_01486]	
[SWS_DM_01487]	
[SWS_DM_01488]	
[SWS_DM_01489]	
[SWS_DM_01490]	
[SWS_DM_01491]	
[SWS_DM_01492]	
[SWS_DM_01493]	
[SWS_DM_01494]	
[SWS_DM_01495]	
[SWS_DM_01496]	
[SWS_DM_01497]	
[SWS_DM_01498]	
[SWS_DM_01499]	
[SWS_DM_01500]	
[SWS_DM_01501]	
[SWS_DM_01502]	
[SWS_DM_01503]	
[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]	





Number	Heading
[SWS_DM_01520]	
[SWS_DM_01521]	
[SWS_DM_01522]	
[SWS_DM_01523]	
[SWS_DM_01524]	
[SWS_DM_01525]	<code>ara::diag::DoIPPowerMode</code> not yet offered when client requests DoIP PowerMode
[SWS_DM_01526]	<code>ara::diag::DoIPActivationLine</code> not yet offered on activation line controlled network interfaces
[SWS_DM_01527]	<code>ara::diag::DoIPGroupIdentification</code> not yet offered when DM needs to retrieve GID
[SWS_DM_01528]	<code>ara::diag::DoIPEntityIdentification</code> not yet offered when DM needs to retrieve EID from Application
[SWS_DM_01529]	Behavior on failed <code>ara::diag</code> 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]	





Number	Heading
[SWS_DM_01556]	
[SWS_DM_01557]	
[SWS_DM_01558]	
[SWS_DM_01559]	
[SWS_DM_01560]	
[SWS_DM_01561]	Response Body forSOVD fault environment_data table

**Table D.22: Added Specification Items in R22-11**

## D.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 <code>UDS</code> service <code>ClearDiagnosticInformation</code>
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00126]	Realisation of <code>UDS</code> service <code>0x3E TesterPresent</code>
[SWS_DM_00128]	Realization of <code>UDS</code> service <code>RequestDownload (0x34)</code>
[SWS_DM_00134]	Realization of <code>UDS</code> service <code>RequestUpload (0x35)</code>
[SWS_DM_00137]	Realization of <code>UDS</code> service <code>TransferData (0x36)</code>
[SWS_DM_00140]	Realisation of <code>UDS</code> service <code>0x28 CommunicationControl</code>
[SWS_DM_00141]	Realization of <code>UDS</code> service <code>RequestTransferExit (0x37)</code>
[SWS_DM_00159]	Allow only to clear <code>GroupOfAllDTCs</code>
[SWS_DM_00160]	Allow to clear single <code>DTCs</code>
[SWS_DM_00170]	Realisation of <code>UDS</code> service <code>ReadDataByIdentifier (0x22)</code>
[SWS_DM_00186]	Realisation of <code>UDS</code> service <code>WriteDataByIdentifier (0x2E)</code>
[SWS_DM_00201]	Realization of <code>UDS</code> service <code>RoutineControl (0x31)</code>
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00213]	<code>DTC</code> 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 <code>UDS</code> service <code>DiagnosticSessionControl</code>
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00229]	Support of <code>UDS</code> service <code>ControlDTCSetting (0x85)</code>





Number	Heading
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00252]	Reaction on Unsupported Subfunction
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00277]	Cancellation of a Diagnostic Conversation in case of External Service Processing
[SWS_DM_00278]	Cancellation of a Diagnostic Conversation in case of Internal Processing
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	
[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_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00363]	Unsupported Subfunction
[SWS_DM_00368]	DM takes care of Response Pending Messages
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19
[SWS_DM_00380]	Support for S3 timer
[SWS_DM_00381]	Session timeout
[SWS_DM_00384]	
[SWS_DM_00412]	Check requested number of DataIdentifiers
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[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]	





Number	Heading
[SWS_DM_00526]	
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[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





Number	Heading
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00578]	
[SWS_DM_00579]	
[SWS_DM_00580]	
[SWS_DM_00581]	
[SWS_DM_00582]	
[SWS_DM_00583]	
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00595]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00600]	
[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]	





Number	Heading
[SWS_DM_00617]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00620]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00634]	
[SWS_DM_00635]	
[SWS_DM_00636]	
[SWS_DM_00637]	
[SWS_DM_00638]	
[SWS_DM_00639]	
[SWS_DM_00640]	
[SWS_DM_00641]	
[SWS_DM_00642]	
[SWS_DM_00643]	
[SWS_DM_00644]	
[SWS_DM_00645]	
[SWS_DM_00646]	
[SWS_DM_00647]	
[SWS_DM_00648]	
[SWS_DM_00649]	
[SWS_DM_00650]	
[SWS_DM_00651]	
[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_00659]	
[SWS_DM_00660]	
[SWS_DM_00661]	
[SWS_DM_00662]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	
[SWS_DM_00666]	
[SWS_DM_00667]	
[SWS_DM_00668]	
[SWS_DM_00669]	
[SWS_DM_00670]	
[SWS_DM_00671]	
[SWS_DM_00672]	
[SWS_DM_00673]	
[SWS_DM_00674]	
[SWS_DM_00690]	
[SWS_DM_00691]	
[SWS_DM_00692]	
[SWS_DM_00693]	
[SWS_DM_00694]	
[SWS_DM_00695]	
[SWS_DM_00696]	
[SWS_DM_00697]	
[SWS_DM_00698]	
[SWS_DM_00699]	
[SWS_DM_00700]	
[SWS_DM_00701]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	
[SWS_DM_00710]	
[SWS_DM_00711]	
[SWS_DM_00712]	
[SWS_DM_00713]	





Number	Heading
[SWS_DM_00714]	
[SWS_DM_00715]	
[SWS_DM_00720]	
[SWS_DM_00721]	
[SWS_DM_00722]	
[SWS_DM_00723]	
[SWS_DM_00724]	
[SWS_DM_00725]	
[SWS_DM_00726]	
[SWS_DM_00730]	
[SWS_DM_00731]	
[SWS_DM_00732]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	
[SWS_DM_00740]	
[SWS_DM_00741]	
[SWS_DM_00742]	
[SWS_DM_00743]	
[SWS_DM_00744]	
[SWS_DM_00745]	
[SWS_DM_00751]	
[SWS_DM_00752]	
[SWS_DM_00753]	
[SWS_DM_00755]	
[SWS_DM_00760]	
[SWS_DM_00761]	
[SWS_DM_00762]	
[SWS_DM_00763]	
[SWS_DM_00764]	
[SWS_DM_00765]	
[SWS_DM_00766]	
[SWS_DM_00767]	
[SWS_DM_00770]	
[SWS_DM_00771]	
[SWS_DM_00772]	
[SWS_DM_00773]	
[SWS_DM_00774]	





Number	Heading
[SWS_DM_00775]	
[SWS_DM_00776]	
[SWS_DM_00777]	
[SWS_DM_00782]	
[SWS_DM_00783]	
[SWS_DM_00784]	
[SWS_DM_00785]	
[SWS_DM_00786]	
[SWS_DM_00787]	
[SWS_DM_00788]	
[SWS_DM_00789]	
[SWS_DM_00790]	
[SWS_DM_00791]	
[SWS_DM_00792]	
[SWS_DM_00793]	
[SWS_DM_00794]	
[SWS_DM_00795]	
[SWS_DM_00797]	
[SWS_DM_00798]	
[SWS_DM_00799]	
[SWS_DM_00800]	
[SWS_DM_00801]	
[SWS_DM_00802]	
[SWS_DM_00803]	
[SWS_DM_00804]	
[SWS_DM_00805]	
[SWS_DM_00806]	
[SWS_DM_00807]	
[SWS_DM_00808]	
[SWS_DM_00809]	
[SWS_DM_00810]	
[SWS_DM_00820]	
[SWS_DM_00821]	
[SWS_DM_00822]	
[SWS_DM_00830]	
[SWS_DM_00831]	
[SWS_DM_00832]	
[SWS_DM_00833]	
[SWS_DM_00834]	





Number	Heading
[SWS_DM_00835]	
[SWS_DM_00836]	
[SWS_DM_00837]	
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00860]	No service processing
[SWS_DM_00863]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00864]	Checking Supported Subfunction for CompareKey
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00933]	UDS DTC status bit 5 / 'testFailedSinceLastClear' after displacement
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00945]	Occurrence Counter initial value
[SWS_DM_00946]	Occurrence Counter increment strategy 'testFailed'-only
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00967]	Support of UDS service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00971]	
[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]	





Number	Heading
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after <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]	





Number	Heading
[SWS_DM_01118]	Support of <a href="#">DEXT</a> parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate
[SWS_DM_01119]	Support of <a href="#">DEXT</a> parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents
[SWS_DM_01120]	Support of <a href="#">DEXT</a> parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of <a href="#">DEXT</a> parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength
[SWS_DM_01122]	Support of <a href="#">DEXT</a> 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]	





Number	Heading
[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]	





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 D.23: Changed Specification Items in R22-11**

### D.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[13]
[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[13]
[SWS_DM_00873]	Diagnostic event processing interface







Number	Heading
[SWS_DM_01084]	Triggering for snapshot record storage (calculated)
[SWS_DM_01091]	Maximum time frame to trigger ExecuteReset()
[SWS_DM_09011]	
[SWS_DM_09013]	
[SWS_DM_09014]	
[SWS_DM_09018]	

**Table D.24: Deleted Specification Items in R22-11**

#### D.8.4 Added Constraints in R22-11

none

#### D.8.5 Changed Constraints in R22-11

none

#### D.8.6 Deleted Constraints in R22-11

none

### D.9 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11

#### D.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





Number	Heading
[SWS_DM_01004]	Definition of API function ara::diag::GetDiagReportingDomain
[SWS_DM_01562]	Definition of API variable ara::diag::CommunicationControl::ComCtrlRequestParamsType::controlType
[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 <a href="#">Transport Protocol Handlers</a>
[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 <a href="#">DiagnosticMemoryDestination.agingRequiresTestedCycle</a>
[SWS_DM_01577]	Behavior of not configured <a href="#">DiagnosticMemoryDestination.statusBitHandlingTestFailedSinceLastClear</a>
[SWS_DM_01578]	Behavior of not configured <a href="#">DiagnosticMemoryDestination.clearDtcLimitation</a>
[SWS_DM_01579]	Behavior of not configured <a href="#">DiagnosticMemoryDestination.memoryEntryStorageTrigger</a>
[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=





Number	Heading
[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
[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=





Number	Heading
[SWS_DM_01649]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01650]	Definition of API function ara::diag::DownloadService::DownloadService
[SWS_DM_01651]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01652]	Definition of API function ara::diag::GenericDataIdentifier::operator=
[SWS_DM_01653]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01654]	Definition of API function ara::diag::GenericDataIdentifier::GenericData Identifier
[SWS_DM_01655]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01656]	Definition of API function ara::diag::GenericUDSService::operator=
[SWS_DM_01657]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01658]	Definition of API function ara::diag::GenericUDSService::Generic UDSService
[SWS_DM_01659]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::operator=
[SWS_DM_01660]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::operator=
[SWS_DM_01661]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface
[SWS_DM_01662]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDEPortInterface::ShortnameOfDEPortInterface
[SWS_DM_01663]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::operator=
[SWS_DM_01664]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::operator=
[SWS_DM_01665]	Definition of API function Namespace1OfPortInterface::Namespace2OfPort Interface::ShortnameOfDIPortInterface::ShortnameOfDIPortInterface
[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=





Number	Heading
[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
[SWS_DM_01703]	Definition of API type ara::diag::EventHandleType
[SWS_DM_01704]	Definition of API class ara::diag::MultipleEvent
[SWS_DM_01705]	Definition of API function ara::diag::MultipleEvent::MultipleEvent
[SWS_DM_01706]	Definition of API function ara::diag::MultipleEvent::~MultipleEvent
[SWS_DM_01707]	Definition of API function ara::diag::MultipleEvent::GetEventStatus
[SWS_DM_01708]	Definition of API function ara::diag::MultipleEvent::SetEventStatusChanged Notifier
[SWS_DM_01709]	Definition of API function ara::diag::MultipleEvent::GetDTCNumber
[SWS_DM_01710]	Definition of API function ara::diag::MultipleEvent::GetDebouncingStatus
[SWS_DM_01711]	Definition of API function ara::diag::MultipleEvent::GetFaultDetectionCounter
[SWS_DM_01726]	Definition of API type ara::diag::ConditionHandleType
[SWS_DM_01727]	Definition of API function ara::diag::MultipleCondition::MultipleCondition
[SWS_DM_01728]	Definition of API function ara::diag::MultipleCondition::GetCondition





Number	Heading
[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
[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





Number	Heading
[SWS_DM_01768]	Reporting 'DTC fault memory cleared' to IdsM
[SWS_DM_01769]	Reporting 'DTC setting switched off' to IdsM
[SWS_DM_01770]	Reporting 'successful authentication' to IdsM
[SWS_DM_01771]	Reporting 'certificate failed' to IdsM
[SWS_DM_01772]	Reporting 'authentication required' to IdsM
[SWS_DM_01773]	Reporting 'required time delay not expired for security access' to IdsM
[SWS_DM_01774]	Reporting 'number of attempts exceeded for security access' to IdsM
[SWS_DM_01775]	Reporting 'security access with invalid key' to IdsM
[SWS_DM_01776]	Reporting 'security unlocked successfully for security access' to IdsM
[SWS_DM_01777]	Reporting 'security access denied' to IdsM
[SWS_DM_01778]	Reporting 'incorrect message length or invalid format' to IdsM
[SWS_DM_01779]	Reporting 'service subfunction not supported' to IdsM
[SWS_DM_01780]	Reporting service not supported to IdsM
[SWS_DM_01781]	(Re-)Identification of Clients by Token
[SWS_DM_01782]	(Re-)Identification of Clients by Identity
[SWS_DM_01783]	Locking only for authorized SOVD clients
[SWS_DM_01784]	Service Validation of SOVD Requests
[SWS_DM_01785]	Service Validation for SOVD Resource Collections
[SWS_DM_01786]	SOVD Online Capability Description role sensitivity
[SWS_DM_01787]	Realization of SOVD Read Configuration
[SWS_DM_01788]	Realization of SOVD Write Configuration
[SWS_DM_01789]	Realization of SOVD API Methods for Handling of Bulk Data
[SWS_DM_01790]	Realization of SOVD Bulk Data API Method for Retrieve List of all Bulk Data Categories
[SWS_DM_01791]	Realization of SOVD Read Bulk Data Meta Data
[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





Number	Heading
[SWS_DM_01806]	Definition of API enum ara::diag::DiagSovdErrc
[SWS_DM_01807]	Definition of API class ara::diag::DiagSovdException
[SWS_DM_01808]	Definition of API function ara::diag::DiagSovdException::DiagSovdException
[SWS_DM_01809]	Definition of API class ara::diag::DiagSovdErrorDomain
[SWS_DM_01810]	Definition of API type ara::diag::DiagSovdErrorDomain::Errc
[SWS_DM_01811]	Definition of API type ara::diag::DiagSovdErrorDomain::Exception
[SWS_DM_01812]	Definition of API function ara::diag::DiagSovdErrorDomain::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







Number	Heading
[SWS_DM_01836]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::file_name
[SWS_DM_01837]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::creation_date
[SWS_DM_01838]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::last_modified
[SWS_DM_01839]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash
[SWS_DM_01840]	Definition of API variable ara::diag::SovdBulkData::BulkDataMetaData Descriptor::hash_algorithm
[SWS_DM_01841]	Definition of API class ara::diag::SovdBulkData
[SWS_DM_01842]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::deleted_items
[SWS_DM_01843]	Definition of API variable ara::diag::SovdBulkData::DeleteByCategory Result::failed_items
[SWS_DM_01844]	Definition of API class ara::diag::SovdBulkData::DeletionError
[SWS_DM_01845]	Definition of API variable ara::diag::SovdBulkData::DeletionError::item
[SWS_DM_01846]	Definition of API variable ara::diag::SovdBulkData::DeletionError::error
[SWS_DM_01847]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01848]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01849]	Definition of API function ara::diag::SovdBulkData::SovdBulkData
[SWS_DM_01850]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01851]	Definition of API function ara::diag::SovdBulkData::operator=
[SWS_DM_01852]	Definition of API function ara::diag::SovdBulkData::~SovdBulkData
[SWS_DM_01853]	Definition of API function ara::diag::SovdBulkData::GetBulkDataMetaData
[SWS_DM_01854]	Definition of API function ara::diag::SovdBulkData::RequestBulkDataUpload
[SWS_DM_01855]	Definition of API function ara::diag::SovdBulkData::DeleteAllBulkData
[SWS_DM_01856]	Definition of API function ara::diag::SovdBulkData::DeleteSpecificBulkData
[SWS_DM_01857]	Definition of API function ara::diag::SovdBulkData::Offer
[SWS_DM_01858]	Definition of API function ara::diag::SovdBulkData::StopOffer
[SWS_DM_01860]	Definition of API class ara::diag::SovdConfiguration
[SWS_DM_01861]	Definition of API class ara::diag::SovdConfiguration::SovdConfigurationMeta Info
[SWS_DM_01862]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::size
[SWS_DM_01863]	Definition of API variable ara::diag::SovdConfiguration::SovdConfiguration MetaInfo::mimetype
[SWS_DM_01864]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01865]	Definition of API function ara::diag::SovdConfiguration::~SovdConfiguration
[SWS_DM_01866]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration
[SWS_DM_01867]	Definition of API function ara::diag::SovdConfiguration::SovdConfiguration





Number	Heading
[SWS_DM_01868]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01869]	Definition of API function ara::diag::SovdConfiguration::operator=
[SWS_DM_01870]	Definition of API function ara::diag::SovdConfiguration::RequestGet Configuration
[SWS_DM_01871]	Definition of API function ara::diag::SovdConfiguration::RequestPut Configuration
[SWS_DM_01872]	Definition of API function ara::diag::SovdConfiguration::Offer
[SWS_DM_01873]	Definition of API function ara::diag::SovdBulkData::RequestBulkData Download
[SWS_DM_01874]	Definition of API class ara::diag::SovdSwUpdate
[SWS_DM_01875]	Definition of API enum ara::diag::SovdUpdatePhase
[SWS_DM_01876]	Definition of API enum ara::diag::SovdUpdateStatus
[SWS_DM_01877]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageDetails
[SWS_DM_01878]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::id
[SWS_DM_01879]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::name
[SWS_DM_01881]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::translation_id
[SWS_DM_01882]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::automated
[SWS_DM_01883]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::origin
[SWS_DM_01884]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes
[SWS_DM_01885]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::notes_translation_id
[SWS_DM_01886]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity
[SWS_DM_01887]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::user_activity_translation_id
[SWS_DM_01888]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions
[SWS_DM_01889]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::preconditions_translation_id
[SWS_DM_01890]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::execution_conditions
[SWS_DM_01891]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::duration
[SWS_DM_01893]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::size
[SWS_DM_01894]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::updated_components





Number	Heading
[SWS_DM_01895]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Details::affected_components
[SWS_DM_01896]	Definition of API class ara::diag::SovdSwUpdate::SubProgress
[SWS_DM_01898]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::entity
[SWS_DM_01899]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::status
[SWS_DM_01900]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::progress
[SWS_DM_01901]	Definition of API variable ara::diag::SovdSwUpdate::SubProgress::error
[SWS_DM_01902]	Definition of API class ara::diag::SovdSwUpdate::UpdatePackageStatus
[SWS_DM_01903]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::phase
[SWS_DM_01904]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::status
[SWS_DM_01905]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::progress
[SWS_DM_01906]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::subprogress
[SWS_DM_01907]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step
[SWS_DM_01908]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::step_translation_id
[SWS_DM_01909]	Definition of API variable ara::diag::SovdSwUpdate::UpdatePackage Status::error
[SWS_DM_01910]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01911]	Definition of API function ara::diag::SovdSwUpdate::~~SovdSwUpdate
[SWS_DM_01912]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01913]	Definition of API function ara::diag::SovdSwUpdate::SovdSwUpdate
[SWS_DM_01914]	Definition of API function ara::diag::SovdSwUpdate::operator=
[SWS_DM_01915]	Definition of API function ara::diag::SovdSwUpdate::operator=
[SWS_DM_01916]	Definition of API function ara::diag::SovdSwUpdate::GetAllUpdates
[SWS_DM_01917]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Details
[SWS_DM_01918]	Definition of API function ara::diag::SovdSwUpdate::PutUpdatePackage Automated
[SWS_DM_01919]	Definition of API function ara::diag::SovdSwUpdate::PrepareUpdatePackage
[SWS_DM_01920]	Definition of API function ara::diag::SovdSwUpdate::ExecuteUpdatePackage
[SWS_DM_01921]	Definition of API function ara::diag::SovdSwUpdate::GetUpdatePackage Status
[SWS_DM_01922]	Definition of API function ara::diag::SovdSwUpdate::DeleteUpdatePackage
[SWS_DM_01923]	Definition of API function ara::diag::SovdSwUpdate::RequestUpdatePackage Registration
[SWS_DM_01924]	Definition of API function ara::diag::SovdSwUpdate::Offer
[SWS_DM_01925]	Definition of API function ara::diag::SovdSwUpdate::StopOffer





Number	Heading
[SWS_DM_01926]	Definition of API class ara::diag::SovdBulkData::DeleteByCategoryResult
[SWS_DM_01927]	Definition of API class ara::diag::SovdAuthorization::ValidateAuthorization Output
[SWS_DM_01928]	SOVD method for logging
[SWS_DM_01929]	SOVD Error for non-reentrant software
[SWS_DM_01930]	Definition of API function ara::diag::SovdConfiguration::StopOffer
[SWS_DM_01931]	
[SWS_DM_01932]	
[SWS_DM_01933]	
[SWS_DM_01934]	
[SWS_DM_01935]	
[SWS_DM_01936]	
[SWS_DM_01937]	
[SWS_DM_01938]	
[SWS_DM_01939]	
[SWS_DM_01940]	
[SWS_DM_01941]	
[SWS_DM_01942]	
[SWS_DM_01943]	
[SWS_DM_01944]	
[SWS_DM_01945]	
[SWS_DM_01946]	
[SWS_DM_01947]	
[SWS_DM_01948]	
[SWS_DM_01949]	
[SWS_DM_09025]	Definition of API variable ara::diag::uds_transport::UdsTransportProtocol Handler::transportprotocolManager

**Table D.25: Added Specification Items in R23-11**

## D.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 <code>kFailed</code>
[SWS_DM_00036]	Debounce timer behavior upon reported <code>kPassed</code>
[SWS_DM_00085]	Internal debounce timer init





Number	Heading
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00224]	Indicator healing
[SWS_DM_00237]	Aging
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00300]	Definition of API function ara::diag::uds_transport::UdsMessage::GetPayload
[SWS_DM_00326]	Definition of API function ara::diag::uds_transport::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 <a href="#">internal DiagnosticDataElements</a>
[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





Number	Heading
[SWS_DM_00597]	Definition of API function Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::Offer
[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





Number	Heading
[SWS_DM_00774]	Definition of API function ara::diag::ServiceValidation::Validate
[SWS_DM_00776]	Definition of API function ara::diag::ServiceValidation::Offer
[SWS_DM_00789]	Definition of API function ara::diag::DownloadService::RequestDownload
[SWS_DM_00790]	Definition of API function ara::diag::DownloadService::DownloadData
[SWS_DM_00791]	Definition of API function ara::diag::DownloadService::RequestDownloadExit
[SWS_DM_00792]	Definition of API function ara::diag::DownloadService::Offer
[SWS_DM_00795]	Definition of API class ara::diag::UploadService::OperationOutput
[SWS_DM_00799]	Definition of API function ara::diag::UploadService::RequestUpload
[SWS_DM_00800]	Definition of API function ara::diag::UploadService::UploadData
[SWS_DM_00801]	Definition of API function ara::diag::UploadService::RequestUploadExit
[SWS_DM_00802]	Definition of API function ara::diag::UploadService::Offer
[SWS_DM_00808]	Definition of API function ara::diag::CommunicationControl::CommCtrl Request
[SWS_DM_00809]	Definition of API function ara::diag::CommunicationControl::Offer
[SWS_DM_00836]	Definition of API function ara::diag::DoIPActivationLine::Offer
[SWS_DM_00877]	Starting time-based event debouncing towards <code>kFinallyDefective</code>
[SWS_DM_00878]	Starting time-based event debouncing towards <code>kFinallyHealed</code>
[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 <code>FDC</code> " as extended data record
[SWS_DM_00952]	Configuration of DTC "max. <code>FDC</code> since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. <code>FDC</code> 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 D.26: Changed Specification Items in R23-11**

### D.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 <a href="#">Aging</a>
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00874]	Reporting <code>kPrepassed</code> or <code>kPrefailed</code> 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</code> method
[SWS_DM_01438]	<a href="#">SOVD</a> data attribute <code>group</code>

**Table D.27: Deleted Specification Items in R23-11**

### D.9.4 Added Constraints in R23-11

none

### D.9.5 Changed Constraints in R23-11

Number	Heading
[SWS_DM_- CONSTR_- 00084]	Each <b>DTC</b> shall be assigned to an event memory destination
[SWS_DM_- CONSTR_- 00960]	No support for <code>DEM_AGINGCTR_UPCNT_FIRST_ACTIVE</code>

**Table D.28: Changed Constraints in R23-11**

### D.9.6 Deleted Constraints in R23-11

Number	Heading
[SWS_DM_- CONSTR_- 00208]	Delay time value for sharedTimer

**Table D.29: Deleted Constraints in R23-11**