

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

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R21-11

Document Change History			
Date	Release	Changed by	Description
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Introduced UDS service 29 • Introduced Event Combination in chapter 7
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Validated requirements from concept DoIPEExtension • Introduced UDS services 2A & 2C
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Document quality improvement and fixing bugs • Incorporated Quality Scope Review Findings • Partly removed obsolete requirements • Removed obsolete service interfaces • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	<ul style="list-style-type: none"> • Document quality improvement and fixing bugs • Introduced ara::diag interfaces in draft state

2018-10-31	18-10	AUTOSAR Release Management	<ul style="list-style-type: none"> • Diagnostic Protocol replaced by Diagnostic Conversations • ResponseOnEvent, CommunicationControl, EcuReset added • Chapter 7 overall rework and updates • Chapter 8 split into chapter 8 (C++ API) and chapter 9 (Service Interfaces)
2018-03-29	18-03	AUTOSAR Release Management	<ul style="list-style-type: none"> • Chapter 7.1. Software Cluster added • Chapter 7.2. Diagnostic Service Management, common parts for all services separated • Chapter 7.3. Event Management, several additions and rework • Chapter 8. API specification, complete rework
2017-10-27	17-10	AUTOSAR Release Management	<ul style="list-style-type: none"> • General API rework • TP Plug-in interface • Introduction of SoftwareCluster in APIs • Additional UDS services like SecurityAccess
2017-03-31	17-03	AUTOSAR Release Management	<ul style="list-style-type: none"> • Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	21
1.1	Diagnostic interface	21
1.2	AUTOSAR Diagnostic Extract Template (DEXT)	21
1.3	Software Cluster	21
1.3.1	Diagnostic Server	21
1.3.2	Diagnostic Managers external dependencies	24
2	Acronyms and Abbreviations	24
3	Related documentation	28
3.1	Input documents & related standards and norms	28
3.2	Further applicable specification	29
4	Constraints and assumptions	29
4.1	Known Limitations	29
5	Dependencies to other modules	31
6	Requirements Tracing	32
6.1	Not applicable requirements	47
7	Functional specification	47
7.1	Functional cluster life-cycle	48
7.2	Startup	48
7.3	Shutdown	49
7.4	UDS Transport Layer	49
7.4.1	Support of proprietary UDS Transport Layer	50
7.4.1.1	Initialization, Starting and Stopping of a proprietary UDS TransportLayer	50
7.4.1.2	UDS message reception on a proprietary UDS TransportLayer	51
7.4.1.3	UDS message transmission on a proprietary UDS TransportLayer	54
7.4.1.4	Channel Notifications	55
7.4.2	DoIP	56
7.4.3	Dispatching of UDS Requests	58
7.5	Diagnostic Server	59
7.5.1	Diagnostic Communication Management	59
7.5.1.1	Diagnostic Conversations	60
7.5.1.1.1	Parallel Client Handling	60
7.5.1.1.2	Life-cycle of a Diagnostic Conversation	63
7.5.1.1.3	Diagnostic Conversation Service Interface	63
7.5.1.2	Assignment of UDS requests to Diagnostic Conversations	65
7.5.1.2.1	Prioritization	67

7.5.1.2.2	Replacement of Diagnostic Conversations and initial values	68
7.5.1.2.3	Refusal of incoming diagnostic request	68
7.5.1.3	Handling Authentication State and DynamicAccessLists	69
7.5.1.3.1	ExternalAuthentication	69
7.5.1.3.2	ClientAuthentication	69
7.5.1.3.3	ClientAuthenticationHandle	71
7.5.1.3.4	DynamicAccessList Creation and Update	72
7.5.1.4	Diagnostic Service Authentication checks	73
7.5.1.5	UDS request Validation/Verification	74
7.5.1.5.1	UDS request format checks	74
7.5.1.5.2	Supported service checks	74
7.5.1.5.3	Session and Security Checks	75
7.5.1.5.4	Manufacturer and Supplier Permission Checks and Confirmation	76
7.5.1.5.5	Condition checks	77
7.5.1.6	UDS response handling	78
7.5.1.6.1	Positive and negative responses	78
7.5.1.6.2	Suppression of responses	78
7.5.1.6.3	Sending busy Responses	78
7.5.1.7	Keep track of active non-default sessions	79
7.5.1.8	UDS service processing	80
7.5.1.8.1	Supported UDS Services	80
7.5.1.8.2	Common service processing items	81
7.5.1.8.3	Service 0x10 – DiagnosticSessionControl	81
7.5.1.8.4	Service 0x11 – ECUReset	82
7.5.1.8.5	Service 0x14 – ClearDiagnosticInformation	84
7.5.1.8.5.1	Clearing user-defined fault memory	85
7.5.1.8.6	Service 0x19 – ReadDTCInformation	86
7.5.1.8.6.1	SF 0x01 – reportNumberOfDTCByStatusMask	87
7.5.1.8.6.2	SF 0x02 – reportDTCByStatusMask	87
7.5.1.8.6.3	SF 0x04 – reportDTCSnapshotRecordByDTCNumber	88
7.5.1.8.6.4	SF 0x06 – reportDTCExtDataRecordByDTCNumber	88
7.5.1.8.6.5	SF 0x07 – reportNumberOfDTCBySeverityMaskRecord	88
7.5.1.8.6.6	SF 0x0A – reportSupportedDTC	89
7.5.1.8.6.7	SF 0x14 – reportDTCFaultDetectionCounter	89
7.5.1.8.6.8	SF 0x17 – reportUserDefMemoryDTCByStatusMask	89
7.5.1.8.6.9	SF 0x18 – reportUserDefMemoryDTCSnapshotRecordByDTCNumber	89

7.5.1.8.6.10	SF 0x19 – reportUserDefMemory-DTCExtDataRecordByDTCNumber	90
7.5.1.8.7	Service 0x22 – ReadDataByIdentifier	90
7.5.1.8.8	Service 0x27 – SecurityAccess	91
7.5.1.8.9	Service 0x28 – CommunicationControl	94
7.5.1.8.10	Service 0x29 – Authentication	94
7.5.1.8.10.1	VerifyCertificateUnidirectional	96
7.5.1.8.10.2	VerifyCertificateBidirectional	97
7.5.1.8.10.3	ProofOfOwnership	98
7.5.1.8.10.4	DeAuthenticate	99
7.5.1.8.10.5	AuthenticationConfiguration	99
7.5.1.8.10.6	TransmitCertificate	99
7.5.1.8.11	Service 0x2A – ReadDataByPeriodicIdentifier	100
7.5.1.8.12	Service 0x2C – DynamicallyDefineDataIdentifier	103
7.5.1.8.13	Service 0x2E – WriteDataByIdentifier	105
7.5.1.8.14	Service 0x31 – RoutineControl	106
7.5.1.8.15	Service 0x34 – RequestDownload	107
7.5.1.8.16	Service 0x35 – RequestUpload	108
7.5.1.8.17	Service 0x36 – TransferData	108
7.5.1.8.18	Service 0x37 – RequestTransferExit	109
7.5.1.8.19	Service 0x3E – TesterPresent	109
7.5.1.8.20	Service 0x85 – ControlDTCSetting	109
7.5.1.8.21	Service 0x86 – ResponseOnEvent	111
7.5.1.8.22	Custom Diagnostic Services	113
7.5.1.9	Cancellation of a Diagnostic Conversation	113
7.5.2	Diagnostic Event Management	115
7.5.2.1	Caching of application call for event processing	115
7.5.2.2	Diagnostic Events	115
7.5.2.2.1	Definition	115
7.5.2.2.2	Monitors	116
7.5.2.2.3	Reporting	118
7.5.2.2.4	Event Combination	119
7.5.2.2.4.1	Combination On Storage	121
7.5.2.2.4.2	Combination On Retrieval	121
7.5.2.2.5	EnableConditions	122
7.5.2.2.6	Debouncing	122
7.5.2.2.6.1	Counter-based debouncing	124
7.5.2.2.6.2	Time-based debouncing	126
7.5.2.2.6.3	Monitor-internal debouncing	129
7.5.2.2.6.4	Debounce algorithm reset	129
7.5.2.2.6.5	Dependencies to enable conditions	130
7.5.2.2.6.6	Dependencies to UDS service 0x85 ControlDTCSettings	130
7.5.2.2.7	Event Status processing	130
7.5.2.2.8	Event status change notifications	132

	7.5.2.2.9	Event occurrence	132
7.5.2.3		Condition Mangement	133
7.5.2.4		Operation Cycles Management	133
7.5.2.5		Event memory	134
	7.5.2.5.1	DTC Introduction	135
	7.5.2.5.1.1	Format	135
	7.5.2.5.1.2	Groups	136
	7.5.2.5.1.3	Priority	136
	7.5.2.5.2	UDS DTC Status	137
	7.5.2.5.2.1	Status processing	137
	7.5.2.5.2.2	UDS DTC Status change notifications	138
	7.5.2.5.2.3	Indicators	138
	7.5.2.5.2.4	User controlled WarningIndicatorRequest-bit	139
	7.5.2.5.3	Destination	140
	7.5.2.5.4	DTC related data	141
	7.5.2.5.4.1	Triggering for data storage	141
	7.5.2.5.4.2	Storage of snapshot record data	141
	7.5.2.5.4.3	Storage of extended data	143
	7.5.2.5.4.4	Internal statistical data elements in EDRs	143
	7.5.2.5.5	Clearing DTCs	147
	7.5.2.5.5.1	Locking of the DTC clearing process by a client	147
	7.5.2.5.5.2	ClearConditions	148
	7.5.2.5.5.3	DTC clearing triggered by application	149
	7.5.2.5.6	Aging	150
	7.5.2.5.7	NumberOfStoredEntries	151
	7.5.2.5.8	Active / Passive Status of Events	151
	7.5.2.5.9	Event memory overflow	151
	7.5.2.5.10	Event memory entry displacement	152
	7.5.2.5.11	Reporting order of event memory entries	156
7.5.3		Required Configuration	157
7.5.4		Diagnostic Data Management	157
	7.5.4.1	Internal and External Diagnostic Data Elements	158
	7.5.4.2	Reading and Writing Diagnostic Data Identifier	160
	7.5.4.2.1	Supported Diagnostic Mappings	160
	7.5.4.2.2	Reading Diagnostic Data Identifier	160
	7.5.4.2.3	Writing Diagnostic Data Identifier	161
8		API specification	163
8.1		C++ language binding <sub component>	163
8.2		API Common Data Types	163
	8.2.1	C++ Diagnostic Error Types	163
8.3		API Reference	178
	8.3.1	C++ Diagnostic API Interfaces	178
	8.3.1.1	Introduction	178

8.3.1.2	MetalInfo class	179
8.3.1.2.1	diag::MetalInfo class	179
8.3.1.2.2	diag::MetalInfo::MetalInfo constructor	179
8.3.1.2.3	diag::MetalInfo::MetalInfo copy constructor	180
8.3.1.2.4	diag::MetalInfo::MetalInfo move constructor	180
8.3.1.2.5	diag::MetalInfo::MetalInfo copy assignment operator	180
8.3.1.2.6	diag::MetalInfo::MetalInfo move assignment operator	181
8.3.1.2.7	diag::MetalInfo::Context enum	181
8.3.1.2.8	diag::MetalInfo::GetValue function	182
8.3.1.2.9	diag::MetalInfo::GetContext function	182
8.3.1.2.10	diag::MetalInfo::~~MetalInfo destructor	182
8.3.1.3	ReentrancyType class	184
8.3.1.3.1	diag::DataIdentifierReentrancyType class	184
8.3.1.4	Monitor class	185
8.3.1.4.1	diag::Monitor::CounterBased type	186
8.3.1.4.2	diag::Monitor::TimeBased type	188
8.3.1.4.3	diag::InitMonitorReason enum	189
8.3.1.4.4	diag::MonitorAction enum	189
8.3.1.4.5	diag::Monitor::Monitor constructors	190
8.3.1.4.6	diag::Monitor::ReportMonitorAction function	191
8.3.1.4.7	diag::Monitor::Offer function	192
8.3.1.4.8	diag::Monitor::StopOffer function	192
8.3.1.5	GenericUDSService class	192
8.3.1.5.1	diag::GenericUDSService::OperationOutput type	193
8.3.1.5.2	diag::GenericUDSService::OperationOutput::responseData	193
8.3.1.5.3	diag::GenericUDSService::GenericUDSService function	193
8.3.1.5.4	diag::GenericUDSService::~~GenericUDSService function	194
8.3.1.5.5	diag::GenericUDSService::Offer function	194
8.3.1.5.6	diag::GenericUDSService::StopOffer function	195
8.3.1.5.7	diag::GenericUDSService::HandleMessage function	195
8.3.1.6	GenericDataIdentifier class	197
8.3.1.6.1	diag::GenericDataIdentifier::OperationOutput type	197
8.3.1.6.2	diag::GenericDataIdentifier::GenericDataIdentifier function	198
8.3.1.6.3	diag::GenericDataIdentifier::~~GenericDataIdentifier function	198
8.3.1.6.4	diag::GenericDataIdentifier::Offer function	199
8.3.1.6.5	diag::GenericDataIdentifier::StopOffer function	199

	8.3.1.6.6	diag::GenericDataIdentifier::Read function	200
	8.3.1.6.7	diag::GenericDataIdentifier::Write function	202
8.3.1.7		GenericRoutine class	204
	8.3.1.7.1	diag::GenericRoutine::OperationOutput type	204
	8.3.1.7.2	diag::GenericRoutine::GenericRoutine function	205
	8.3.1.7.3	diag::GenericRoutine::~~GenericRoutine function	205
	8.3.1.7.4	diag::GenericRoutine::Offer function	205
	8.3.1.7.5	diag::GenericRoutine::StopOffer function	206
	8.3.1.7.6	diag::GenericRoutine::Start function	206
	8.3.1.7.7	diag::GenericRoutine::Stop function	208
	8.3.1.7.8	diag::GenericRoutine::RequestResults function	210
8.3.1.8		CancellationHandler class	212
	8.3.1.8.1	diag::CancellationHandler::CancellationHandler function	213
	8.3.1.8.2	diag::CancellationHandler::IsCanceled function	214
	8.3.1.8.3	diag::CancellationHandler::SetNotifier function	215
8.3.1.9		Event class	215
	8.3.1.9.1	diag::DTCFormatType type	215
	8.3.1.9.2	diag::EventStatusBit type	216
	8.3.1.9.3	diag::Event::EventStatusByte type	216
	8.3.1.9.4	diag::Event::DebouncingState type	217
	8.3.1.9.5	diag::Event::Event constructor	217
	8.3.1.9.6	diag::Event::~~Event destructor	217
	8.3.1.9.7	diag::Event::GetEventStatus function	218
	8.3.1.9.8	diag::Event::SetEventStatusChangedNotifier function	218
	8.3.1.9.9	diag::Event::GetLatchedWIRStatus function	218
	8.3.1.9.10	diag::Event::SetLatchedWIRStatus function	219
	8.3.1.9.11	diag::Event::GetDTCNumber function	219
	8.3.1.9.12	diag::Event::GetDebouncingStatus function	220
	8.3.1.9.13	diag::Event::GetTestComplete function	220
	8.3.1.9.14	diag::Event::GetFaultDetectionCounter function	220
8.3.1.10		DTCInformation class	221
	8.3.1.10.1	diag::ControlDtcStatusType type	221
	8.3.1.10.2	diag::UdsDtcStatusBitType type	221
	8.3.1.10.3	diag::DTCInformation::UdsDtcStatusByteType type	222
	8.3.1.10.4	diag::DTCInformation::SnapshotDataIdentifierType type	222
	8.3.1.10.5	diag::DTCInformation::SnapshotDataRecordType type	223
	8.3.1.10.6	diag::DTCInformation::SnapshotRecordUpdatedType type	223
	8.3.1.10.7	diag::DTCInformation::DTCInformation function	223
	8.3.1.10.8	diag::DTCInformation::~~DTCInformation function	224

8.3.1.10.9	diag::DTCInformation::GetCurrentStatus function	224
8.3.1.10.10	diag::DTCInformation::SetDTCStatusChangedNotifier function	224
8.3.1.10.11	diag::DTCInformation::SetSnapshotRecordUpdatedNotifier function	225
8.3.1.10.12	diag::DTCInformation::GetNumberOfStoredEntries function	225
8.3.1.10.13	diag::DTCInformation::SetNumberOfStoredEntriesNotifier function	226
8.3.1.10.14	diag::DTCInformation::Clear function	226
8.3.1.10.15	diag::DTCInformation::GetControlDTCStatus function	227
8.3.1.10.16	diag::DTCInformation::SetControlDtcStatusNotifier function	227
8.3.1.10.17	diag::DTCInformation::EnableControlDtc function	227
8.3.1.10.18	diag::DTCInformation::GetEventMemoryOverflow function	228
8.3.1.10.19	diag::DTCInformation::SetEventMemoryOverflowNotifier function	228
8.3.1.11	Conversation class	228
8.3.1.11.1	diag::ActivityStatusType type	229
8.3.1.11.2	diag::SessionControlType type	229
8.3.1.11.3	diag::SecurityLevelType type	230
8.3.1.11.4	diag::Conversation::ConversationIdentifierType type	230
8.3.1.11.5	diag::Conversation::GetConversation function	230
8.3.1.11.6	diag::Conversation::GetAllConversations function	231
8.3.1.11.7	diag::Conversation::GetCurrentActiveConversations function	231
8.3.1.11.8	diag::Conversation::GetActivityStatus function	232
8.3.1.11.9	diag::Conversation::SetActivityNotifier function	232
8.3.1.11.10	diag::Conversation::GetConversationIdentifier function	232
8.3.1.11.11	diag::Conversation::GetDiagnosticSession function	233
8.3.1.11.12	diag::Conversation::SetDiagnosticSessionNotifier function	233
8.3.1.11.13	diag::Conversation::GetDiagnosticSessionShortName function	233
8.3.1.11.14	diag::Conversation::GetDiagnosticSecurityLevel function	234
8.3.1.11.15	diag::Conversation::SetSecurityLevelNotifier function	234

8.3.1.11.16	diag::Conversation::GetDiagnosticSecurityLevelShortName function	234
8.3.1.11.17	diag::Conversation::ResetToDefaultSession function	235
8.3.1.12	Condition class	235
8.3.1.12.1	diag::ConditionType type	236
8.3.1.12.2	diag::Condition::Condition function	236
8.3.1.12.3	diag::Condition::~~Condition function	236
8.3.1.12.4	diag::Condition::GetCondition function	237
8.3.1.12.5	diag::Condition::SetCondition function	237
8.3.1.13	OperationCycle class	237
8.3.1.13.1	diag::OperationCycle::OperationCycle function	238
8.3.1.13.2	diag::OperationCycle::~~OperationCycle function	238
8.3.1.13.3	diag::OperationCycle::SetNotifier function	238
8.3.1.13.4	diag::OperationCycle::RestartOperationCycle function	239
8.3.1.14	Indicator class	239
8.3.1.14.1	diag::IndicatorType type	239
8.3.1.14.2	diag::Indicator::Indicator function	240
8.3.1.14.3	diag::Indicator::~~Indicator function	240
8.3.1.14.4	diag::Indicator::GetIndicator function	240
8.3.1.14.5	diag::Indicator::SetNotifier function	241
8.3.1.15	ServiceValidation class	241
8.3.1.15.1	diag::ConfirmationStatusType	242
8.3.1.15.2	diag::ServiceValidation::ServiceValidation function	242
8.3.1.15.3	diag::ServiceValidation::~~ServiceValidation function	243
8.3.1.15.4	diag::ServiceValidation::Validate function	243
8.3.1.15.5	diag::ServiceValidation::Confirmation function	245
8.3.1.15.6	diag::ServiceValidation::Offer function	245
8.3.1.15.7	diag::ServiceValidation::StopOffer function	246
8.3.1.16	SecurityAccess class	246
8.3.1.16.1	diag::KeyCompareResultType type	247
8.3.1.16.2	diag::SecurityAccess::SecurityAccess function	247
8.3.1.16.3	diag::SecurityAccess::~~SecurityAccess function	247
8.3.1.16.4	diag::SecurityAccess::GetSeed function	248
8.3.1.16.5	diag::SecurityAccess::CompareKey function	248
8.3.1.16.6	diag::SecurityAccess::Offer function	249
8.3.1.16.7	diag::SecurityAccess::StopOffer function	250
8.3.1.17	CommunicationControl class	250
8.3.1.17.1	diag::CommunicationControl::ComCtrlRequestParamsType type	250
8.3.1.17.2	diag::CommunicationControl::CommunicationControl function	251

8.3.1.17.3	diag::CommunicationControl::~~CommunicationControl function	251
8.3.1.17.4	diag::CommunicationControl::CommCtrlRequest function	251
8.3.1.17.5	diag::CommunicationControl::Offer function	252
8.3.1.17.6	diag::CommunicationControl::StopOffer function	252
8.3.1.18	DownloadService class	253
8.3.1.18.1	diag::DownloadService::OperationOutput type	253
8.3.1.18.2	diag::DownloadService::DownloadService function	254
8.3.1.18.3	diag::DownloadService::~~DownloadService function	254
8.3.1.18.4	diag::DownloadService::RequestDownload function	255
8.3.1.18.5	diag::DownloadService::DownloadData function	255
8.3.1.18.6	diag::DownloadService::RequestDownloadExit function	256
8.3.1.18.7	diag::DownloadService::Offer function	257
8.3.1.18.8	diag::DownloadService::StopOffer function	257
8.3.1.19	UploadService class	258
8.3.1.19.1	diag::UploadService::OperationOutput type	258
8.3.1.19.2	diag::UploadService::UploadService function	259
8.3.1.19.3	diag::UploadService::~~UploadService function	259
8.3.1.19.4	diag::UploadService::RequestUpload function	259
8.3.1.19.5	diag::UploadService::UploadData function	260
8.3.1.19.6	diag::UploadService::RequestUploadExit function	261
8.3.1.19.7	diag::UploadService::Offer function	262
8.3.1.19.8	diag::UploadService::StopOffer function	262
8.3.1.20	EcuResetRequest class	263
8.3.1.20.1	diag::EcuResetRequest::ResetRequestType type	263
8.3.1.20.2	diag::EcuResetRequest::StopOffer function	263
8.3.1.20.3	diag::EcuResetRequest::Offer function	264
8.3.1.20.4	diag::EcuResetRequest::RequestReset function	264
8.3.1.20.5	diag::EcuResetRequest::ExecuteReset function	265
8.3.1.20.6	diag::EcuResetRequest::EnableRapidShutdown function	265
8.3.1.20.7	diag::EcuResetRequest::~~EcuResetRequest destructor	266
8.3.1.20.8	diag::EcuResetRequest::EcuResetRequest constructor	266
8.3.1.21	Authentication class	266
8.3.1.21.1	diag::Authentication::Authentication constructor	267
8.3.1.21.2	diag::Authentication::~~Authentication destructor	267

8.3.1.21.3	diag::Authentica- tion::VerifyCertificateUnidirectional function . . .	268
8.3.1.21.4	diag::Authentica- tion::VerifyCertificateBidirectional function . . .	269
8.3.1.21.5	diag::Authentication::VerifyOwnership function	270
8.3.1.21.6	diag::Authentication::TransmitCertificate function	271
8.3.1.21.7	diag::Authentication::Offer function	272
8.3.1.21.8	diag::Authentication::StopOffer function	272
8.3.1.22	ClientAuthentication class	272
8.3.1.22.1	diag::ClientAuthentication::DiagnosticAuthState type	273
8.3.1.22.2	diag::ClientAuthentication::DiagnosticAuthRole type	273
8.3.1.22.3	diag::ClientAuthentication::~~ClientAuthentication destructor	273
8.3.1.22.4	diag::ClientAuthentication::ClientAuthentication copy constructor	274
8.3.1.22.5	diag::ClientAuthentication::ClientAuthentication move constructor	274
8.3.1.22.6	diag::ClientAuthentication::ClientAuthentication copy assignment operator	274
8.3.1.22.7	diag::ClientAuthentication::ClientAuthentication move assignment operator	275
8.3.1.22.8	diag::ClientAuthentica- tion::OverrideDefaultRoles function	275
8.3.1.22.9	diag::ClientAuthentication::Authenticate function	275
8.3.1.22.10	diag::ClientAuthentication::GetState function .	276
8.3.1.22.11	diag::ClientAuthentication::SetNotifier function	276
8.3.1.23	ClientAuthenticationHandle class	277
8.3.1.23.1	diag::ClientAuthenticationHandle::ClientAu- thenticationHandle constructor	277
8.3.1.23.2	diag::ClientAuthenticationHandle::~~ClientAu- thenticationHandle destructor	277
8.3.1.23.3	diag::ClientAuthenticationHandle::ClientAu- thenticationHandle copy constructor	278
8.3.1.23.4	diag::ClientAuthenticationHandle::ClientAu- thenticationHandle move constructor	278
8.3.1.23.5	diag::ClientAuthenticationHandle::ClientAu- thenticationHandle copy assignment operator	279
8.3.1.23.6	diag::ClientAuthenticationHandle::ClientAu- thenticationHandle move assignment operator	279
8.3.1.23.7	diag::ClientAuthenticationHandle::Append function	279
8.3.1.23.8	diag::ClientAuthenticationHandle::Set function	280
8.3.1.23.9	diag::ClientAuthenticationHandle::Revoke function	280

	8.3.1.23.10	diag::ClientAuthenticationHandle::Refresh function	281
8.3.1.24		DiagnosticServiceDynamicAccessList class	281
	8.3.1.24.1	diag::DiagnosticServiceDynamicAccessList:: DiagnosticServiceDynamicAccessList con- structor	281
	8.3.1.24.2	diag::DiagnosticServiceDynamicAccessList:: ~DiagnosticServiceDynamicAccessList de- structor	282
	8.3.1.24.3	diag::DiagnosticServiceDynamicAccessList:: DiagnosticServiceDynamicAccessList copy constructor	282
	8.3.1.24.4	diag::DiagnosticServiceDynamicAccessList:: DiagnosticServiceDynamicAccessList move constructor	282
	8.3.1.24.5	diag::DiagnosticServiceDynamicAccessList:: DiagnosticServiceDynamicAccessList copy assignment operator	283
	8.3.1.24.6	diag::DiagnosticServiceDynamicAccessList:: DiagnosticServiceDynamicAccessList move assignment operator	283
	8.3.1.24.7	diag::DiagnosticServiceDynamicAccessList:: Reserve function	284
	8.3.1.24.8	diag::DiagnosticServiceDynamicAccessList:: MakeServiceBuilder function	284
	8.3.1.24.9	diag::DiagnosticServiceDynamicAccessList:: MakeServiceBuilder function	284
8.3.1.25		ExternalAuthentication class	285
	8.3.1.25.1	diag::ExternalAuthentication::Address type	285
	8.3.1.25.2	diag::ExternalAuthentica- tion::ExternalAuthentication constructor	286
	8.3.1.25.3	diag::ExternalAuthentica- tion::ExternalAuthentication copy constructor	286
	8.3.1.25.4	diag::ExternalAuthentica- tion::ExternalAuthentication move constructor	286
	8.3.1.25.5	diag::ExternalAuthentica- tion::ExternalAuthentication copy assignment operator	287
	8.3.1.25.6	diag::ExternalAuthentica- tion::ExternalAuthentication move assignment operator	287
	8.3.1.25.7	diag::ExternalAuthentica- tion::~~ExternalAuthentication destructor	287
	8.3.1.25.8	diag::ExternalAuthentication::Get function	288
	8.3.1.25.9	diag::ExternalAuthentication::Get function	288
	8.3.1.25.10	diag::ExternalAuthentication::GetAll function	289

8.3.1.26	DynamicAccessListDiagServiceBuilder class	289
8.3.1.26.1	ByteRange class	289
8.3.1.26.1.1	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange constructor	290
8.3.1.26.1.2	diag::DynamicAccessListDiagServiceBuilder::ByteRange::~~ByteRange destructor	290
8.3.1.26.1.3	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy constructor	291
8.3.1.26.1.4	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move constructor	291
8.3.1.26.1.5	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy assignment operator	291
8.3.1.26.1.6	diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move assignment operator	292
8.3.1.26.1.7	diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin function	292
8.3.1.26.1.8	diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax function	293
8.3.1.26.2	diag::DynamicAccessListDiagServiceBuilder::Byte type	293
8.3.1.26.3	diag::DynamicAccessListDiagServiceBuilder::ByteString type	293
8.3.1.26.4	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder constructor	294
8.3.1.26.5	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy constructor	294
8.3.1.26.6	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move constructor	294
8.3.1.26.7	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy assignment operator	295
8.3.1.26.8	diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move assignment operator	295
8.3.1.26.9	diag::DynamicAccessListDiagServiceBuilder::~~DynamicAccessListDiagServiceBuilder destructor	296

8.3.1.26.10	diag::DynamicAccessListDiagService-Builder::Add function	296
8.3.1.26.11	diag::DynamicAccessListDiagService-Builder::Add function	296
8.3.1.26.12	diag::DynamicAccessListDiagService-Builder::Add function	297
8.3.1.26.13	diag::DynamicAccessListDiagService-Builder::Any function	297
8.3.1.26.14	diag::DynamicAccessListDiagService-Builder::EndsWith function	298
8.3.1.26.15	diag::DynamicAccessListDiagService-Builder::EndsWith function	298
8.3.1.26.16	diag::DynamicAccessListDiagService-Builder::Build function	299
8.3.2	C++ Diagnostic generated API Interfaces	299
8.3.2.1	Implementation Types header files	300
8.3.2.2	Typed Routine class	300
8.3.2.2.1	Routine::StartOutput type	301
8.3.2.2.2	Routine::StopOutput type	301
8.3.2.2.3	Routine::RequestResultsOutput type	301
8.3.2.2.4	Routine constructor	302
8.3.2.2.5	Routine destructor	302
8.3.2.2.6	Routine::Offer function	302
8.3.2.2.7	Routine::StopOffer function	303
8.3.2.2.8	Routine::Start function	303
8.3.2.2.9	Routine::Stop function	305
8.3.2.2.10	Routine::RequestResults function	307
8.3.2.3	Typed DataIdentifier class	310
8.3.2.3.1	DataIdentifier::OperationOutput type	310
8.3.2.3.2	DataIdentifier constructor	310
8.3.2.3.3	DataIdentifier destructor	311
8.3.2.3.4	DataIdentifier::Offer function	311
8.3.2.3.5	DataIdentifier::StopOffer function	311
8.3.2.3.6	DataIdentifier::Read function	312
8.3.2.3.7	DataIdentifier::Write function	314
8.3.2.4	Typed DataElement class	316
8.3.2.4.1	DataElement::OperationOutput type	316
8.3.2.4.2	DataElement constructor	317
8.3.2.4.3	DataElement destructor	317
8.3.2.4.4	DataElement::Offer function	317
8.3.2.4.5	DataElement::StopOffer function	318
8.3.2.4.6	DataElement::Read function	318
A	Mentioned Manifest Elements	320
B	Platform Extension API (normative)	373
B.1	C++ UDS Transportlayer API Interfaces	374

B.1.1	UDS Transportlayer Types	374
B.1.1.1	uds_transport::ByteVector	374
B.1.1.2	uds_transport::ChannelID	374
B.1.1.3	uds_transport::Priority	374
B.1.1.4	uds_transport::ProtocolKind	375
B.1.1.5	uds_transport::UdsMessageConstPtr	375
B.1.1.6	uds_transport::UdsMessagePtr	376
B.1.1.7	uds_transport::UdsTransportProtocolHandlerID	376
B.1.2	UdsMessage Class	376
B.1.2.1	Types	377
B.1.2.1.1	uds_transport::UdsMessage::Address	377
B.1.2.1.2	uds_transport::UdsMessage::MetaInfoMap	377
B.1.2.1.3	uds_transport::UdsMessage::TargetAddressType	377
B.1.2.2	Methods	378
B.1.2.2.1	uds_transport::UdsMessage::UdsMessage	378
B.1.2.2.2	uds_transport::UdsMessage::UdsMessage	378
B.1.2.2.3	uds_transport::UdsMessage::UdsMessage	379
B.1.2.2.4	uds_transport::UdsMessage::UdsMessage::operator=	379
B.1.2.2.5	uds_transport::UdsMessage::UdsMessage::operator=	379
B.1.2.2.6	uds_transport::UdsMessage::~~UdsMessage	380
B.1.2.2.7	uds_transport::UdsMessage::AddMetaInfo	380
B.1.2.2.8	uds_transport::UdsMessage::GetPayload	381
B.1.2.2.9	uds_transport::UdsMessage::GetSa	381
B.1.2.2.10	uds_transport::UdsMessage::GetTa	382
B.1.2.2.11	uds_transport::UdsMessage::GetTaType	382
B.1.3	UdsTransportProtocolHandler Class	383
B.1.3.1	Types	383
B.1.3.1.1	uds_transport::UdsTransportProtocolHandler::InitializationResult	383
B.1.3.2	Methods	383
B.1.3.2.1	uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler	383
B.1.3.2.2	uds_transport::UdsTransportProtocolHandler::~~UdsTransport	384
B.1.3.2.3	uds_transport::UdsTransportProtocolHandler::GetHandlerID	384
B.1.3.2.4	uds_transport::UdsTransportProtocolHandler::Initialize	385
B.1.3.2.5	uds_transport::UdsTransportProtocolHandler::NotifyReestablishment	385
B.1.3.2.6	uds_transport::UdsTransportProtocolHandler::Start	386

	B.1.3.2.7	uds_transport::UdsTransportProtocolHandler:: Stop	386
	B.1.3.2.8	uds_transport::UdsTransportProtocolHandler:: Transmit	387
	B.1.3.2.9	uds_transport::UdsTransportProtocolHandler:: GetPeriodicHandler	387
B.1.4		UdsTransportProtocolMgr Class	388
	B.1.4.1	Types	388
	B.1.4.1.1	uds_transport::UdsTransportProtocolMgr:: GlobalChannelIdentifier	388
	B.1.4.1.2	uds_transport::UdsTransportProtocolMgr:: IndicationResult	388
	B.1.4.1.3	uds_transport::UdsTransportProtocolMgr:: TransmissionResult	389
	B.1.4.2	Methods	389
	B.1.4.2.1	uds_transport::UdsTransportProtocolMgr:: ChannelReestablished	389
	B.1.4.2.2	uds_transport::UdsTransportProtocolMgr:: HandleMessage	390
	B.1.4.2.3	uds_transport::UdsTransportProtocolMgr:: HandlerStopped	390
	B.1.4.2.4	uds_transport::UdsTransportProtocolMgr:: IndicateMessage	391
	B.1.4.2.5	uds_transport::UdsTransportProtocolMgr:: NotifyMessageFailure	391
	B.1.4.2.6	uds_transport::UdsTransportProtocolMgr:: TransmitConfirmation	392
	B.1.4.2.7	uds_transport::UdsTransportProtocolMgr:: PeriodicTransmitConfirmation	392
B.1.5		UdsTransportProtocolPeriodicHandler Class	393
	B.1.5.1	Methods	393
	B.1.5.1.1	uds_transport::UdsTransportProtocolPeri- odicHandler::GetNumberOfPeriodicMessages	393
	B.1.5.1.2	uds_transport::UdsTransportProtocolPeri- odicHandler::GetMaxPayloadLength	393
	B.1.5.1.3	uds_transport::UdsTransportProtocolPeri- odicHandler::PeriodicTransmit	394
B.1.6		Sequence Diagrams of UDS Transport Layer Interaction	395
	B.1.6.1	Lifecycle	395
	B.1.6.2	UDS Request Processing	397
	B.1.6.3	UDS Response Transmission	399
	B.1.6.4	Channel Reestablishment	401
B.2		C++ DoIP API Interfaces	402
	B.2.1	DoIPGroupIdentification class	402
	B.2.1.1	diag::DoIPGroupIdentification::DoIPGroupIdentifica- tionType type	402

B.2.1.2	diag::DoIPGroupIdentification::DoIPGroupIdentifica- tion function	402
B.2.1.3	diag::DoIPGroupIdentification::~~DoIPGroupIdentifi- cation function	403
B.2.1.4	diag::DoIPGroupIdentification::GetGidStatus function	403
B.2.1.5	diag::DoIPGroupIdentification::Offer function	403
B.2.1.6	diag::DoIPGroupIdentification::StopOffer function . .	404
B.2.2	DoIPPowerMode class	404
B.2.2.1	diag::PowerModeType type	405
B.2.2.2	diag::DoIPPowerMode::DoIPPowerMode function . .	405
B.2.2.3	diag::DoIPPowerMode::~~DoIPPowerMode function .	405
B.2.2.4	diag::DoIPPowerMode::GetDoIPPowerMode function	406
B.2.2.5	diag::DoIPPowerMode::Offer function	406
B.2.2.6	diag::DoIPPowerMode::StopOffer function	406
B.2.3	DoIPActivationLine class	407
B.2.3.1	diag::DoIPActivationLine::DoIPActivationLine function	407
B.2.3.2	diag::DoIPActivationLine::~~DoIPActivationLine function	408
B.2.3.3	diag::DoIPActivationLine::GetNetworkInterfaceld function	408
B.2.3.4	diag::DoIPActivationLine::UpdateActivationLineState function	408
B.2.3.5	diag::DoIPActivationLine::GetActivationLineState function	409
B.2.3.6	diag::DoIPActivationLine::Offer function	409
B.2.3.7	diag::DoIPActivationLine::StopOffer function	409
B.2.4	DoIPTriggerVehicleAnnouncement class	410
B.2.4.1	diag::DoIPTriggerVehicleAnnouncement::GetDoIP TriggerVehicleAnnouncement function	410
B.2.4.2	diag::DoIPTriggerVehicleAnnouncement::TriggerVe- hicleAnnouncement function	411
C	Interfaces to other Functional Clusters (informative)	411
C.1	Overview	411
C.2	Interface Tables	412
D	History of Constraints and Specification Items	412
D.1	Constraint and Specification Item History of this document according to AUTOSAR Release 17-10	412
D.1.1	Added Traceables in 17-10	412
D.1.2	Changed Traceables in 17-10	415
D.1.3	Deleted Traceables in 17-10	417
D.1.4	Added Constraints in 17-10	417
D.1.5	Changed Constraints in 17-10	417
D.1.6	Deleted Constraints in 17-10	417
D.2	Constraint and Specification Item History of this document according to AUTOSAR Release 18-03	418
D.2.1	Added Traceables in 18-03	418

D.2.2	Changed Traceables in 18-03	419
D.2.3	Deleted Traceables in 18-03	425
D.2.4	Added Constraints in 18-03	425
D.2.5	Changed Constraints in 18-03	425
D.2.6	Deleted Constraints in 18-03	426
D.3	Constraint and Specification Item History of this document according to AUTOSAR Release 18-10	426
D.3.1	Added Traceables in 18-10	426
D.3.2	Changed Traceables in 18-10	428
D.3.3	Deleted Traceables in 18-10	434
D.3.4	Added Constraints in 18-10	435
D.3.5	Changed Constraints in 18-10	435
D.3.6	Deleted Constraints in 18-10	435
D.4	Constraint and Specification Item History of this document according to AUTOSAR Release 19-03	436
D.4.1	Added Traceables in 19-03	436
D.4.2	Changed Traceables in 19-03	441
D.4.3	Deleted Traceables in 19-03	442
D.4.4	Added Constraints in 19-03	442
D.4.5	Changed Constraints in 19-03	442
D.4.6	Deleted Constraints in 19-03	442
D.5	Constraint and Specification Item History of this document according to AUTOSAR Release R19-11	443
D.5.1	Added Traceables in 19-11	443
D.5.2	Changed Traceables in 19-11	446
D.5.3	Deleted Traceables in 19-11	451
D.5.4	Added Constraints in 19-11	454
D.5.5	Changed Constraints in 19-11	454
D.5.6	Deleted Constraints in 19-11	454
D.6	Constraint and Specification Item History of this document according to AUTOSAR Release R20-11	454
D.6.1	Added Traceables in R20-11	454
D.6.2	Changed Traceables in R20-11	460
D.6.3	Deleted Traceables in R20-11	463
D.6.4	Added Constraints in R20-11	464
D.6.5	Changed Constraints in R20-11	464
D.6.6	Deleted Constraints in R20-11	464
D.7	Constraint and Specification Item History of this document according to AUTOSAR Release R21-11	464
D.7.1	Added Traceables in R21-11	464
D.7.2	Changed Traceables in R21-11	469
D.7.3	Deleted Traceables in R21-11	479
D.7.4	Added Constraints in R21-11	480
D.7.5	Changed Constraints in R21-11	480
D.7.6	Deleted Constraints in R21-11	480

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] for the Autosar Adaptive Platform. Unless stated otherwise in this document, the [DM](#) implements the functionality as defined in the ISO 14229-1[1]. Derivations, limitation, OEM or supplier-specific behaviour according to ISO 14229-1[1] are described in this document.

In general the ISO release 2013 [1] is referenced. Only in certain cases, the ISO release 2020 [2] is referenced. This is marked by dedicated bibliographic links to this standard.

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](#).

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]. That means:

- Receiving UDS diagnostic request messages from the network layer
- Extracting transport layer independent UDS information from it.
- Dispatching the request towards the Diagnostic Server instances depending on target address and target address type (physical or functional) of received UDS request message
- Correlating the diagnostic request to an existing UDS session (if already exists)
- Checking whether the diagnostic request is allowed within current session and security settings
- If diagnostic request is NOT allowed, generate negative UDS response and send it to the network layer
- If diagnostic 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.

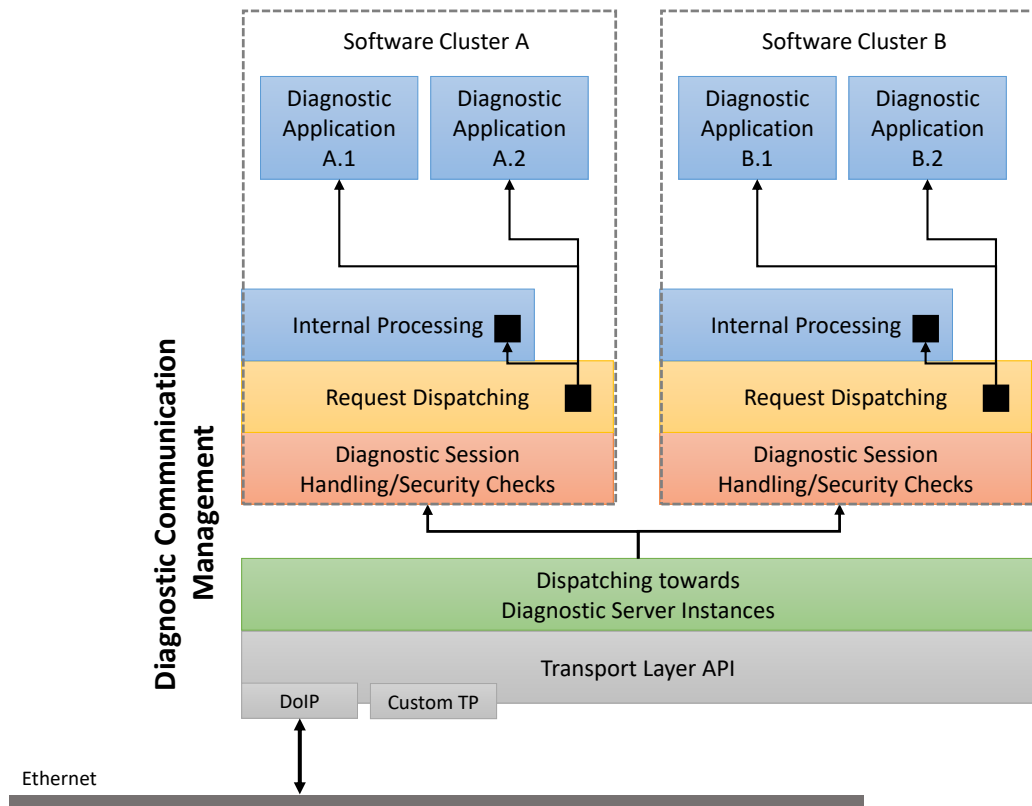


Figure 1.1: Architecture Diagnostic Communication Management

1.3.2 Diagnostic Managers external dependencies

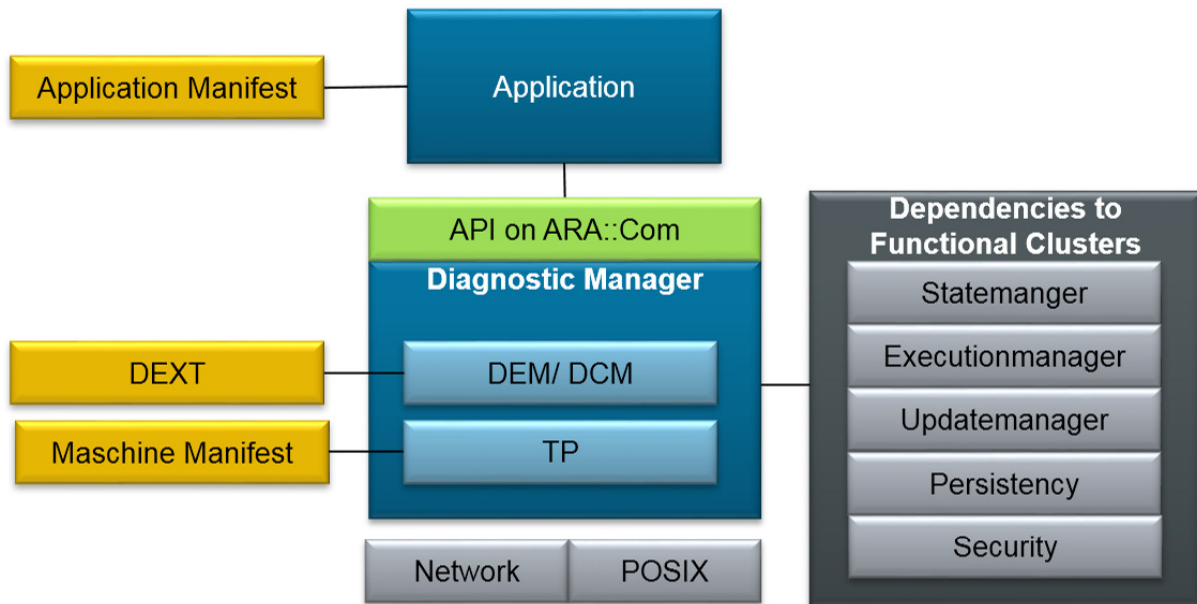


Figure 1.2: Diagnostic Managers external dependencies

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
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
DEXT	AUTOSAR Diagnostic Extract[3], describing diagnostic configuration of an ECU
DM	AUTOSAR Adaptive Diagnostic Management
DTC	Diagnostic Trouble Code according to ISO 14229-1[1]
DDID	Dynamically Defined Data Identifier according to ISO 14229-1[1].

Abbreviation / Acronym:	Description:
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.
ECU	Electronic control unit
EDR	Extended Data Record
Execution Management	Functional cluster Execution Management
FDC	Fault Detection Counter according to ISO 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
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 vehicle's 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[2].
PowerMode	Vehicle basic status information retrieval of DoIP
SA	SourceAddress of a UDS request
SID	Service Identifier, identifying a diagnostic service according to UDS, such as 0x14 ClearDiagnosticInformation
TA	TargetAddress of a UDS request
UDS	Unified Diagnostic Services
VIN	Vehicle Identification Number according to ISO-3779
Dcm	Diagnostic Communication Manager (Module of the AUTOSAR Classic Platform)
DoIP	Diagnostics over Internet Protocol (Communication protocol of automotive electronics according to ISO-13400[5])

Terms:	Description:
Aging	Unlearning/deleting of a no longer failed event/DTC after a defined number of operation cycles from event memory.
Diagnostic Client	A Diagnostic Client is a diagnostic service requester, i.e. sends a UDS request to the Diagnostic Server. Usually the Diagnostic Client is an external tester equipment but can also be another vehicle internal ECU.
Diagnostic Communication Management	Diagnostic Communication Management is the part of the Diagnostic Management which belongs to tester communication and the processing of UDS services.
Diagnostic Conversation	Diagnostic Conversation represents a conversation between Diagnostic Client (Tester) and Diagnostic Server.
Diagnostic Event Management	Diagnostic Event Management is the part of the Diagnostic Management which belongs to processing and storing of diagnostic events and associated data.
Diagnostic Management	Diagnostic Management is a placeholder for the complete functionality of diagnostic communication and event handling.

Terms:	Description:
Diagnostic Server instance	Diagnostic Server (DM) is intended to support an own Diagnostic Server instance per installed <i>SoftwareCluster</i> , see section 7.5 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 <i>DiagnosticServiceClass</i> and the <i>DiagnosticAccessPermission</i> , see 7.5.1.5.3 for a detailed description.
Displacement	In case of an <i>Event memory overflow</i> : 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 <i>DTCs</i> . 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 <i>DTCStatusAvailabilityMask</i> - byte is used in UDS responses to requests for certain sub-functions of service 0x19. It express which of the <i>UDS DTC status bits</i> are supported by the DM for masking purposes.
Enable Conditions	The criteria / conditions under which the test results from the <i>monitors</i> in the <i>AA</i> 's are valid and shall be processed by <i>DM</i> . Configuration is done per <i>event</i> .
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 <i>Event</i> level. Contains the following bits: Nr. Definition: <ul style="list-style-type: none"> • 0 testFailed • 1 testFailedThisOperationCycle • 6 testNotCompletedThisOperationCycle Compare <i>UDS DTC status bit</i>
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 <i>DiagnosticDataElement</i> is either managed internally inside <i>DM</i> or by an external adaptive applications, see 7.5.4.1 for the precise definition.

Terms:	Description:
Internally, Externally	Definition of the support type of a SID by the DM. Internally means processing is done by DM itself, Externally means an external service processor is used.
internal data element	A DiagnosticDataElement which is provided by the DM itself. See also 7.5.4.1 .
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 event .
Operation cycle	A new operation cycle is the start of a new monitoring cycle. This is reflected in a reset of the <code>testFailedThisOperationCycle</code> and <code>testNotCompletedThisOperationCycle</code> bits in the DTC status and optional notification to the monitor to restart the monitoring.
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.
Snapshot Record	Set of measurement values stored in the fault memory at a certain point of time during fault detection. It is used to gain environmental data information for occurred faults.
SoftwareCluster	A <code>SoftwareCluster</code> groups all AUTOSAR artifacts which are relevant to deploy software on a machine. This includes the definition of applications, i.e. their executables, application manifests, communication and diagnostics. In the context of diagnostics a <code>SoftwareCluster</code> can be addressed individually by its own set of diagnostic addresses.
SourceAddress	A Source Address is used to encode client and server identifiers. In a UDS request the source address encodes the Diagnostic Client whereas the source address in a UDS response encodes the Diagnostic Server.
TargetAddress	A Target Address is used to encode client and server identifiers. In a UDS request the target address encodes the Diagnostic Server whereas the target address in a UDS response encodes the Diagnostic Client .
Transport Protocol Handler	A subcomponent of <code>DM</code> implementing a particular Transport Protocol (either <code>DoIP</code> 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].

Terms:	Description:
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 DTC. The following eight bits are defined:</p> <p>Nr: Definition:</p> <ul style="list-style-type: none"> 0 testFailed 1 testFailedThisOperationCycle 2 pendingDTC 3 confirmedDTC 4 testNotCompletedSinceLastClear 5 testFailedSinceLastClear 6 testNotCompletedThisOperationCycle 7 warningIndicatorRequested <p>All eight bits constitute the UDS DTC status byte. Compare Event status</p>
UDS DTC status byte	<p>Bit-packed DTC status information byte as defined in ISO 14229-1[1], based on DTC level. Contains the UDS DTC status bits.</p>
User-defined event memory	<p>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.</p>
Non-volatile Memory	<p>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.</p>

3 Related documentation

3.1 Input documents & related standards and norms

- [1] Unified diagnostic services (UDS) – Part 1: Specification and requirements (Release 2013-03)
<http://www.iso.org>
- [2] Unified diagnostic services (UDS) – Part 1: Application layer (Release 2020-02)
<http://www.iso.org>
- [3] Diagnostic Extract Template
 AUTOSAR_TPS_DiagnosticExtractTemplate
- [4] Glossary
 AUTOSAR_TR_Glossary

- [5] Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
<http://www.iso.org>
- [6] Specification of Adaptive Platform Core
AUTOSAR_SWS_AdaptivePlatformCore
- [7] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement
- [8] Specification of Log and Trace
AUTOSAR_SWS_LogAndTrace
- [9] Specification of Persistency
AUTOSAR_SWS_Persistency
- [10] Requirements on Diagnostics
AUTOSAR_RS_Diagnostics
- [11] Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2: Network layer services
- [12] Road vehicles – Diagnostic communication over Internet Protocol (DoIP) – Part 2: Network and transport layer requirements and services
<http://www.iso.org>
- [13] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification
- [14] Unified diagnostic services (UDS) - Part 2: Session layer services (Release 2013-03)
<http://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 [2] 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
 - 0x38 RequestFileTransfer
 - 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 0x03 (`reportDTCSnapshotIdentification`)
 - 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 [2]), 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.

5 Dependencies to other modules

As any other process started by `Execution Management` [7], `DM` needs to interact with the `Execution Management`.

The `DM` may use `ara::log` ([8], Log and Trace) for logging and tracing purposes.

`DM` may use `ara::per` ([9], Persistency) to store non-volatile data.

6 Requirements Tracing

The following tables reference the requirements specified in [10] 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]

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]
[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]
[RS_AP_00128]	Error reporting.	[SWS_DM_00544]
[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]
[RS_AP_00133]	noexcept behavior of move and swap operations	[SWS_DM_00610] [SWS_DM_00612]
[RS_AP_00134]	noexcept behavior of class destructors	[SWS_DM_00553] [SWS_DM_00584] [SWS_DM_00586] [SWS_DM_00588] [SWS_DM_00590] [SWS_DM_00635] [SWS_DM_00648] [SWS_DM_00665] [SWS_DM_00713] [SWS_DM_00723] [SWS_DM_00733] [SWS_DM_00743] [SWS_DM_00753] [SWS_DM_00763] [SWS_DM_00773] [SWS_DM_00788] [SWS_DM_00798] [SWS_DM_00807] [SWS_DM_00832]

Requirement	Description	Satisfied by
[RS_AP_00137]	Connecting run-time interface with model.	[SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00552] [SWS_DM_00585] [SWS_DM_00587] [SWS_DM_00589] [SWS_DM_00616] [SWS_DM_00631] [SWS_DM_00633] [SWS_DM_00634] [SWS_DM_00647] [SWS_DM_00664] [SWS_DM_00712] [SWS_DM_00722] [SWS_DM_00732] [SWS_DM_00742] [SWS_DM_00752] [SWS_DM_00762] [SWS_DM_00772] [SWS_DM_00787] [SWS_DM_00797] [SWS_DM_00806]
[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]
[RS_AP_00139]	Return type of synchronous function calls.	[SWS_DM_00543] [SWS_DM_00594] [SWS_DM_00597] [SWS_DM_00599] [SWS_DM_00619] [SWS_DM_00638] [SWS_DM_00649] [SWS_DM_00650] [SWS_DM_00651] [SWS_DM_00652] [SWS_DM_00653] [SWS_DM_00654] [SWS_DM_00655] [SWS_DM_00656] [SWS_DM_00666] [SWS_DM_00667] [SWS_DM_00668] [SWS_DM_00669] [SWS_DM_00670] [SWS_DM_00671] [SWS_DM_00672] [SWS_DM_00673] [SWS_DM_00674] [SWS_DM_00692] [SWS_DM_00694] [SWS_DM_00695] [SWS_DM_00696] [SWS_DM_00697] [SWS_DM_00698] [SWS_DM_00699] [SWS_DM_00700] [SWS_DM_00714] [SWS_DM_00715] [SWS_DM_00725] [SWS_DM_00735] [SWS_DM_00744] [SWS_DM_00745] [SWS_DM_00755] [SWS_DM_00766] [SWS_DM_00776] [SWS_DM_00792] [SWS_DM_00802] [SWS_DM_00809] [SWS_DM_00918] [SWS_DM_00919] [SWS_DM_01016] [SWS_DM_01088] [SWS_DM_01102]

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_CONSTR_00208]
[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_00365] [SWS_DM_00433] [SWS_DM_00862]
[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_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_04059]	Configuration of timing parameters	[SWS_DM_NA]
[RS_Diag_04063]	Process a dedicated event identifier for each monitoring path to support an autonomous handling of different events/ faults	[SWS_DM_00007]
[RS_Diag_04064]	Provide configurable buffer sizes for storage of the events, status information and environmental data	[SWS_DM_00920]

Requirement	Description	Satisfied by
[RS_Diag_04067]	Provide the diagnostic status information according to ISO 14229-1	[SWS_DM_00061] [SWS_DM_00062] [SWS_DM_00063] [SWS_DM_00213] [SWS_DM_00217] [SWS_DM_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]
[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_00040] [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_00874] [SWS_DM_00875] [SWS_DM_00876] [SWS_DM_00879] [SWS_DM_00880] [SWS_DM_00952] [SWS_DM_01099] [SWS_DM_01100] [SWS_DM_01101]
[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]
[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_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]

Requirement	Description	Satisfied by
[RS_Diag_04109]	Provide an interface to retrieve the number of event memory entries	[SWS_DM_00669] [SWS_DM_00670] [SWS_DM_00902]
[RS_Diag_04115]	The optional parameter DTCSettingControlOption Record 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_CONSTR_00961]
[RS_Diag_04119]	Handle the execution of diagnostic services according to the assigned diagnostic session	[SWS_DM_00046]
[RS_Diag_04120]	Support a predefined Address AndLengthFormatIdentifier	[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]
[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]
[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_00242] [SWS_DM_CONSTR_00960]
[RS_Diag_04136]	Configurable "confirmed" threshold	[SWS_DM_00218]
[RS_Diag_04140]	Aging for UDS status bits "confirmedDTC" and "testFailed SinceLastClear"	[SWS_DM_00243]

Requirement	Description	Satisfied by
[RS_Diag_04147]	Communication with the transport layers to receive and send diagnostic data	[SWS_DM_00291] [SWS_DM_00293] [SWS_DM_00294] [SWS_DM_00296] [SWS_DM_00297] [SWS_DM_00298] [SWS_DM_00299] [SWS_DM_00300] [SWS_DM_00301] [SWS_DM_00302] [SWS_DM_00303] [SWS_DM_00304] [SWS_DM_00306] [SWS_DM_00307] [SWS_DM_00309] [SWS_DM_00310] [SWS_DM_00311] [SWS_DM_00312] [SWS_DM_00313] [SWS_DM_00314] [SWS_DM_00315] [SWS_DM_00319] [SWS_DM_00322] [SWS_DM_00323] [SWS_DM_00325] [SWS_DM_00326] [SWS_DM_00327] [SWS_DM_00336] [SWS_DM_00337] [SWS_DM_00338] [SWS_DM_00384] [SWS_DM_00451] [SWS_DM_00452] [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]
[RS_Diag_04148]	Provide capabilities to inform applications about diagnostic data changes	[SWS_DM_00667] [SWS_DM_00894]
[RS_Diag_04150]	Support the primary fault memory defined by ISO 14229-1	[SWS_DM_00055] [SWS_DM_00056] [SWS_DM_00083] [SWS_DM_00657] [SWS_DM_00664] [SWS_DM_00911] [SWS_DM_CONSTR_00084]
[RS_Diag_04151]	Event status handling	[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]
[RS_Diag_04157]	Reporting of DTCs and related data	[SWS_DM_00058] [SWS_DM_00061] [SWS_DM_00062] [SWS_DM_00063] [SWS_DM_00218] [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]
[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]

Requirement	Description	Satisfied by
[RS_Diag_04160]	ResponseOnEvent according to ISO 14229-1	[SWS_DM_00491] [SWS_DM_00493] [SWS_DM_00494] [SWS_DM_01098] [SWS_DM_01117] [SWS_DM_01118] [SWS_DM_01119] [SWS_DM_01120] [SWS_DM_01121] [SWS_DM_01122]
[RS_Diag_04166]	Several tester conversations in parallel with assigned priorities	[SWS_DM_00425] [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]
[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_00482] [SWS_DM_00577] [SWS_DM_00847] [SWS_DM_00984]
[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_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]

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]
[RS_Diag_04171]	Synchronous and asynchronous interaction with external service processors	[SWS_DM_NA]
[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_CONSTR_00168]
[RS_Diag_04179]	Provide interfaces for monitoring application.	[SWS_DM_00007] [SWS_DM_00540] [SWS_DM_00541] [SWS_DM_00542] [SWS_DM_00543] [SWS_DM_00548] [SWS_DM_00549] [SWS_DM_00550] [SWS_DM_00873] [SWS_DM_00965]
[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_00218] [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]

Requirement	Description	Satisfied by
		[SWS_DM_00968] [SWS_DM_01028]
[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]
[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]
[RS_Diag_04189]	Support a fine grained configuration for Snapshot Records and ExtendedData Records	[SWS_DM_00151] [SWS_DM_00155]
[RS_Diag_04190]	Usage of internal data elements in SnapshotRecords and ExtendedDataRecords	[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]
[RS_Diag_04194]	ClearDTC shall be accessible for applications	[SWS_DM_00262] [SWS_DM_00671] [SWS_DM_00897] [SWS_DM_00898] [SWS_DM_00899] [SWS_DM_00900] [SWS_DM_00901]
[RS_Diag_04195]	Chronological reporting order of the DTCs located in the configured event memory	[SWS_DM_00981] [SWS_DM_00982] [SWS_DM_NA]
[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_00114] [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_00361]

Requirement	Description	Satisfied by
		[SWS_DM_00363] [SWS_DM_00376] [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_00797] [SWS_DM_00798] [SWS_DM_00799] [SWS_DM_00800] [SWS_DM_00801] [SWS_DM_00802] [SWS_DM_00803] [SWS_DM_00804] [SWS_DM_00805] [SWS_DM_00806] [SWS_DM_00807] [SWS_DM_00808] [SWS_DM_00809] [SWS_DM_00810] [SWS_DM_00860] [SWS_DM_00861] [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_01091] [SWS_DM_01092]
[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_00857] [SWS_DM_00858] [SWS_DM_00970]
[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_NA]
[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_CONSTR_00059]
[RS_Diag_04202]	No description	[SWS_DM_NA]

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_00221] [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]
[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_01084] [SWS_DM_01085] [SWS_DM_01086] [SWS_DM_01087]
[RS_Diag_04206]	Support of ExtendedData Records	[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]
[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 Read DataByPeriodicIdentifier (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]

Requirement	Description	Satisfied by
		[SWS_DM_01064] [SWS_DM_01065] [SWS_DM_01066] [SWS_DM_01067] [SWS_DM_01068] [SWS_DM_01069]
[RS_Diag_04216]	Support for multiple Diagnostic Server Instances	[SWS_DM_00390] [SWS_DM_00391] [SWS_DM_00420]
[RS_Diag_04218]	Support of UDS service 0x2F InputOutputControlByIdentifier	[SWS_DM_NA]
[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]
[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_00040] [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_00879] [SWS_DM_00880]
[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 DynamicallyDefineDataIdentifier (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_01232] [SWS_DM_01233] [SWS_DM_01234] [SWS_DM_01235] [SWS_DM_01236] [SWS_DM_01237] [SWS_DM_01238] [SWS_DM_01239] [SWS_DM_01240] [SWS_DM_01241] [SWS_DM_01242] [SWS_DM_01243]

Requirement	Description	Satisfied by
		[SWS_DM_01244] [SWS_DM_01245] [SWS_DM_01246] [SWS_DM_01247] [SWS_DM_01248] [SWS_DM_01249] [SWS_DM_01250] [SWS_DM_01251]
[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_CONSTR_00206]
[SRS_Eth_00027]	DoIP diagnostic message shall have a format	[SWS_DM_00449]
[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]
[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_00005] [SWS_DM_00449]
[SRS_Eth_00084]	DoIP shall implement routing activation mechanism	[SWS_DM_00449]

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_04202](#), [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 [7.4](#) for details. The subcomponent of [DM](#) implementing a particular Transport Protocol is called a [Transport Protocol Handler](#).

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]. 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 7.5.

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.

A broad subcomponent view on DM is given as follows:

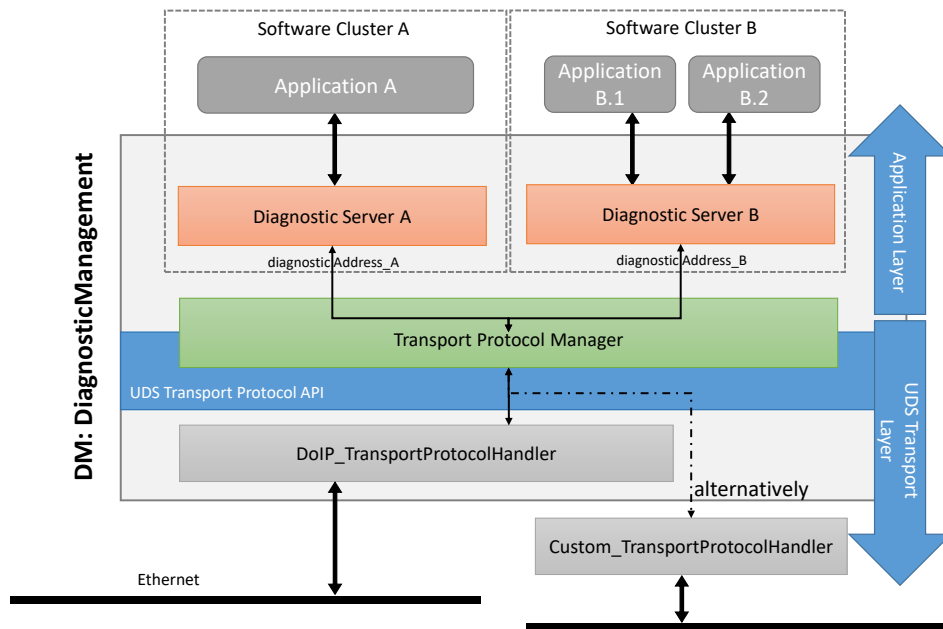


Figure 7.1: Component view on Diagnostic Management

7.1 Functional cluster life-cycle

No content.

7.2 Startup

No content.

7.3 Shutdown

No content.

7.4 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 'Not Acknowledge' (NACK) codes in that case, while CAN (Controller Area Network) does not have any TP failures.

7.4.1 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].

7.4.1.1 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` method of the UDS Transport Protocol implementation during startup/initialization phase, before reporting `ApplicationState.kRunning` to the execution management.](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_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_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*)

7.4.1.2 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[11]) 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::ByteVector` 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 `suppressPosRspMsgIndicationBit = TRUE` are exceptional requests in UDS and valid at any point in time. Therefore the DM allows to check for functional received `TesterPresent`

request in the `ara::diag::uds_transport::UdsTransportProtocolMgr::IndicateMessage`. For handling of the functional `TesterPresent`, see chapter 7.5.1.8.19.

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::UdsTransportProtocolMgr::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](#))

7.4.1.3 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*)

7.4.1.4 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*)

7.4.2 DoIP

[SWS_DM_00005]{DRAFT} DoIP Support [The **DM** shall implement/provide a UDS Transport Layer implementation on Ethernet compliant with ISO-13400[5], also called DoIP.] (*SRS_Eth_00083*)

[SWS_DM_00475]{DRAFT} DoIP Version [The **DM** shall support the following version of the DoIP ISO 13400-2 specification: 2020.] (*RS_Diag_04242*)

Note: According to the ISO 13400-2[12] specification, the DoIP entity shall support 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 7.1.] (*SRS_Eth_00026, SRS_Eth_00080, SRS_Eth_00082, SRS_Eth_00083, SRS_Eth_00084, SRS_Eth_00027*)

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

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_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_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`.

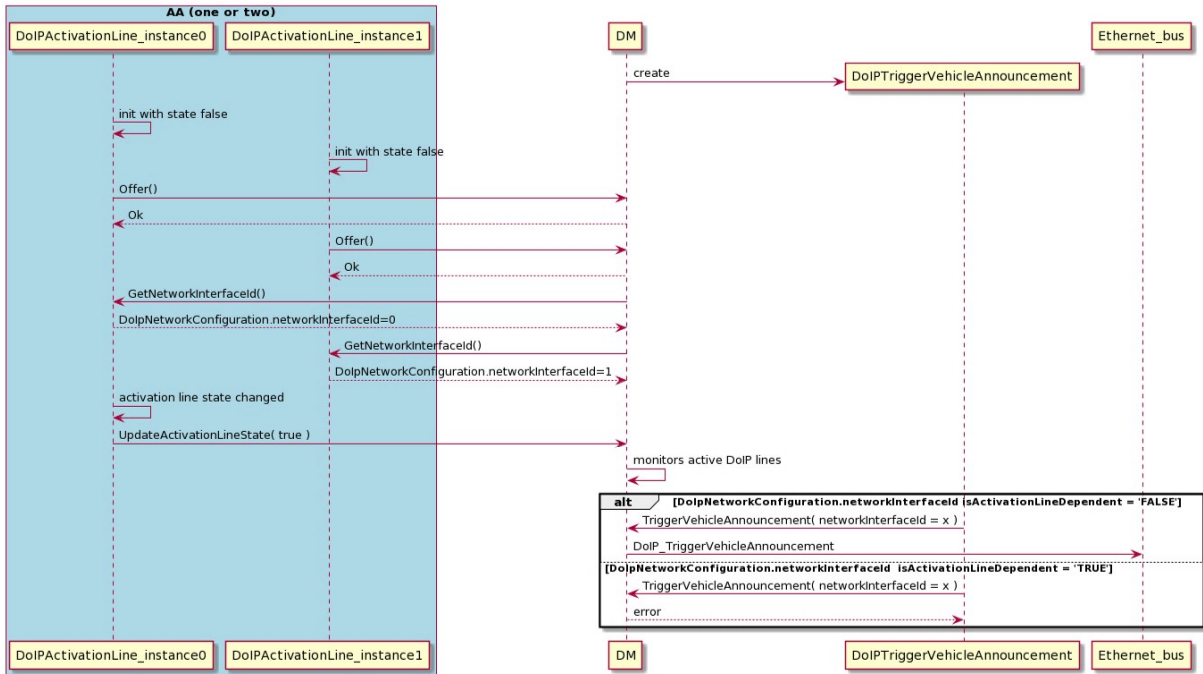


Figure 7.2: Example Sequence Diagram for Activation Line and Vehicle Announcement API use case

7.4.3 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](#))

7.5 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), each [SoftwareCluster](#) has its own `diagnosticAddresses`. For details on the semantics and precise configuration of [SoftwareClusters](#), see [13].

DM is intended to support an own [Diagnostic Server instance](#) per installed [SoftwareCluster](#). 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 configured [SoftwareCluster](#) which references a [DiagnosticContributionSet](#) 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 7.5.3.

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 [SoftwareClusters](#) shall be processed independently by the respective [Diagnostic Servers](#).

7.5.1 Diagnostic Communication Management

A central element in the handling of diagnostic communication is the term [Diagnostic Conversation](#), which is described in section 7.5.1.1. A UDS request is always processed in the context of a [Diagnostic Conversation](#). A single [Diagnostic Server](#) can handle multiple [Diagnostic Conversations](#) in parallel.

7.5.1.1 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*)

7.5.1.1.1 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.

[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::DataIdentifierReentrancyType` 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::DataIdentifierReentrancyType` 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::DataIdentifierReentrancyType` 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.

7.5.1.1.2 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 7.5.1.9) 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 according to [SWS_DM_00365],
 - suppressing negative response according to [SWS_DM_00862].
 - 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.

7.5.1.1.3 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)

7.5.1.2 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.

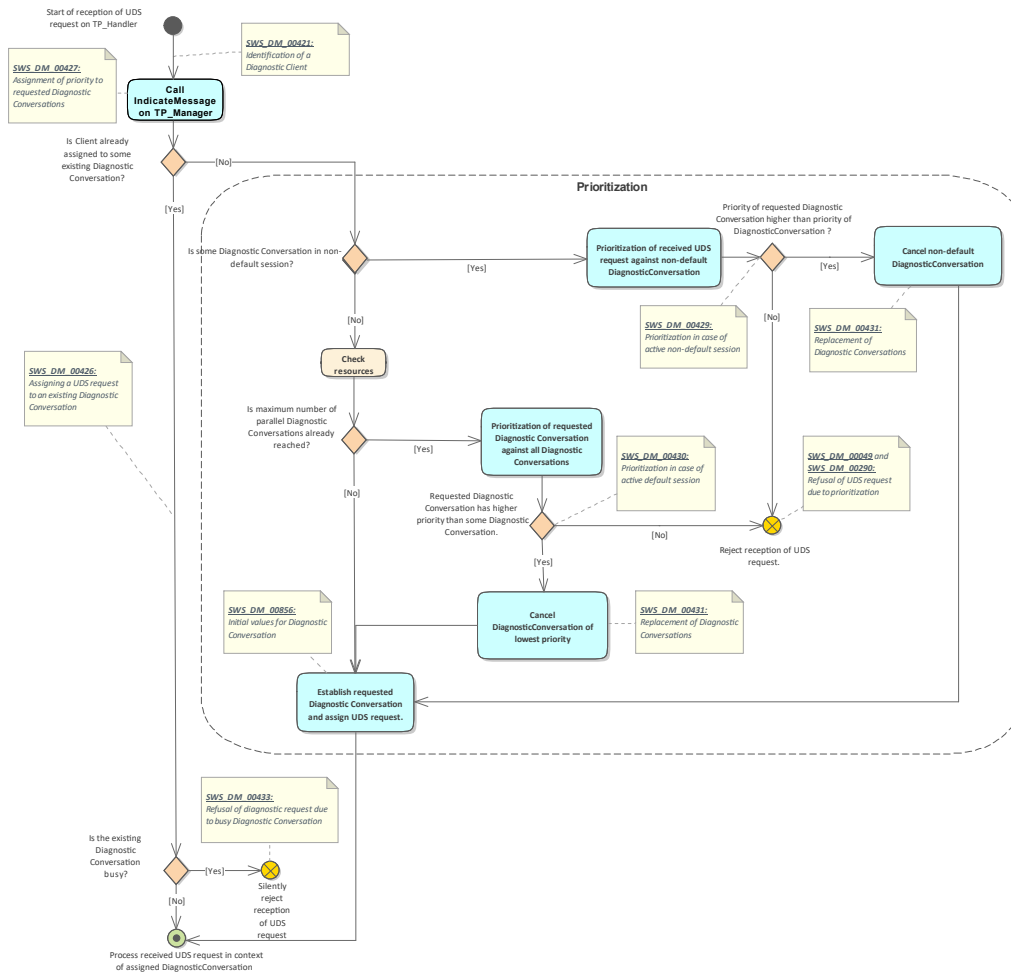


Figure 7.3: UDS request assignment to a Diagnostic Conversation and Prioritization

[SWS_DM_00425]{DRAFT} Procedure to assign UDS requests to Diagnostic Conversations [The *Diagnostic Server instance* shall handle a newly received UDS request as specified in Figure 7.3.] (*RS_Diag_04166*)

[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 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].

7.5.1.2.1 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 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*)

7.5.1.2.2 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_00482] 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*)

7.5.1.2.3 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*)

7.5.1.3 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.

7.5.1.3.1 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`.

[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](#))

7.5.1.3.2 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 Conversation`.

[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 of the class `ara::diag::ClientAuthentication`, 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_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 [In Default Session, if a client is in `kAuthenticated` state, and no diagnostic request is received from the client for a time period `DiagnosticAuthentication.authenticationTimeout`, the Diagnostic Server instance shall set the Authentication State of the client to `kDeAuthenticated`.]([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 of all clients to `kDeAuthenticated`.]([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](#))

7.5.1.3.3 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_01214]{DRAFT} Default DynamicAccessList [On startup, the DynamicAccessList of all clients shall be empty.] ([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](#))

7.5.1.3.4 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_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 `DIDs` 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::DynamicAccessListDiagServiceBuilder::EndsWith` of the class `ara::diag::DynamicAccessListDiagServiceBuilder` are called by the application, the `Diagnostic Server` instance shall add the requested pattern to the end of the `DynamicAccessList`.] ([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*].

7.5.1.4 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_01223]{DRAFT} Diagnostic service role verification [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.DiagnosticAuthRole`. 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` (Refer [*SWS_DM_01213*]).] (*RS_Diag_04233*)

[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*)

7.5.1.5 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 [7.5.1.5.4](#)
- **SID supported?** Decision according to section [7.5.1.5.2](#)
- **SID supported in active session?** Decision according to section [7.5.1.5.3](#)
- **SID security check o.k.?** Decision according to section [7.5.1.5.3](#)
- **supplier-specific failure detected?** Decision by applying supplier-specific checks according to section [7.5.1.5.4](#)

[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.

7.5.1.5.1 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](#))

7.5.1.5.2 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_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*)

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

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

the Verification shall be considered as failed and the negative response code shall be 0x33 (`securityAccessDenied`).] (*RS_Diag_04203*)

7.5.1.5.4 Manufacturer and Supplier Permission Checks and Confirmation

[SWS_DM_00857]{DRAFT} Signature of Manufacturer Permission Check Method [The `Diagnostic Server instance` shall call `ara::diag::ServiceValidation::Validate`, according to its modeled instance given by [TPS_MANI_01311] on each received request message. In case a call returned an error, the Verification shall be considered as failed and the negative response code shall be equal to the value of the error code according to [SWS_DM_00547].] (*RS_Diag_04199*)

[SWS_DM_00858]{DRAFT} Signature of Supplier Permission Check Method [The `Diagnostic Server instance` shall call `ara::diag::ServiceValidation::Validate`, according to its modeled instance given by [TPS_MANI_01312] on each received request message. In case a call returned an error, the Verification shall be considered as failed and the negative response code shall be equal to the value of the error code according to [SWS_DM_00547].] (*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 Manufacturer- or Supplier Permission Check (according to [SWS_DM_00857] or [SWS_DM_00858]) returns the error code `kNoProcessingNoResponse`, the `Diagnostic Server instance` shall call without any service processing the `ara::diag::ServiceValidation::Confirmation` with `ara::diag::ConfirmationStatusType` status parameter set to `kNoProcessingNoResponse` and do no response message.] (*RS_Diag_04196*)

7.5.1.5.5 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*)

7.5.1.6 UDS response handling

7.5.1.6.1 Positive and negative responses

[SWS_DM_00376]{DRAFT} Positive response processing [If an external service processor did not raise an `ApApplicationError`, the `Diagnostic Server instance` shall return a positive response.]([RS_Diag_04196](#))

[SWS_DM_00861]{DRAFT} Negative response processing [If the external processor raised an error according to [\[SWS_DM_00547\]](#), the `Diagnostic Server instance` shall return a negative response with the value of the error code. For details see ISO 14229-1[1]; chapter 10.2.]([RS_Diag_04196](#))

7.5.1.6.2 Suppression of responses

[SWS_DM_00365]{DRAFT} Suppression of positive response in accordance to ISO 14229-1[1] [In the case that the "suppressPosRspMsgIndicationBit" is set in the request, the `Diagnostic Server instance` shall suppress the positive response.]([RS_Diag_04020](#))

[SWS_DM_00862]{DRAFT} Suppression of negative response for functional requests in accordance to ISO 14229-1[1] [If the external processor raised an error according to [\[SWS_DM_00547\]](#), the `Diagnostic Server instance` shall suppress a negative response for the following error codes:

- `kServiceNotSupported`,
- `kSubfunctionNotSupported`,
- `kRequestOutOfRange`,
- `kServiceNotSupportedInActiveSession` or
- `kSubFunctionNotSupportedInActiveSession`

and the request is functional addressed.]([RS_Diag_04020](#))

7.5.1.6.3 Sending busy Responses

[SWS_DM_00368]{DRAFT} Sending busy responses [If the `Diagnostic Server instance` is able to perform a diagnostic service, but needs additional time to finish the task and prepare the response, then the `Diagnostic Server instance` shall send a negative response with NRC 0x78 (`requestCorrectlyReceived-ResponsePending`) when reaching the response time (`p2ServerMax/p2StarServerMax`).]([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*)

7.5.1.7 Keep track of active non-default sessions

[SWS_DM_00380]{DRAFT} Support for S3 timer [The `Diagnostic Server instance` shall provide support for `S3Server` (session timeout) with a fixed value of 5 seconds. The timer handling shall be implemented according to ISO 14229-2[14].] (*RS_Diag_04006*)

[SWS_DM_00381]{DRAFT} Session timeout [Whenever a non-default session is active and when the session timeout (`S3Server`) 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[14].] (*RS_Diag_04006*)

[SWS_DM_00382]{DRAFT} Session timeout start [The session timeout timer (`S3server`) 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 `S3Server` means reset the timer and start counting from the beginning.] (*RS_Diag_04006*)

[SWS_DM_00383]{DRAFT} Session timeout stop [The session timeout timer (`S3Server`) 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*)

7.5.1.8 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](#))

7.5.1.8.1 Supported UDS Services

The [Diagnostic Server instance](#) shall support the following listed UDS services:

SID	Service	Support Type	Reference
0x10	DiagnosticSessionControl	Internally	7.5.1.8.3
0x11	ECUReset	Externally	7.5.1.8.4
0x14	ClearDiagnosticInformation	Internally	7.5.1.8.5
0x19	ReadDTCInformation	Internally	7.5.1.8.6
0x22	ReadDataByIdentifier	Internally & Externally	7.5.1.8.7
0x27	SecurityAccess	Internally & Externally	7.5.1.8.8
0x28	CommunicationControl	Externally	7.5.1.8.9
0x2E	WriteDataByIdentifier	Externally	7.5.1.8.13
0x31	RoutineControl	Externally	7.5.1.8.14
0x34	RequestDownload	Externally	7.5.1.8.15
0x35	RequestUpload	Externally	7.5.1.8.16
0x36	TransferData	Externally	7.5.1.8.17
0x37	RequestTransferExit	Externally	7.5.1.8.18
0x3E	TesterPresent	Internally	7.5.1.8.19
0x85	ControlDTCSetting	Internally	7.5.1.8.20
0x86	ResponseOnEvent	Internally	7.5.1.8.21

Table 7.2: UDS Services supported by [Diagnostic Server instance](#)

Note:

- UDS services which are not supported by DM, are documented in the section [Known Limitations](#).
- Support Type [Internally](#) means, that the service with the given SID can be completely processed internally within the [Diagnostic Server instance](#) without relying on external functionality - typically in form of an AA. Support Type [Externally](#) means, that the [Diagnostic Server instance](#) needs to call an external function, to be able to process the service with the given SID. The mixed support Type "Internally & Externally" means, that for the service with the given SID partially calls to an external function have to be done, but it partially could be also handled internally.

7.5.1.8.2 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](#))

7.5.1.8.3 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)*

7.5.1.8.4 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 `Diagnostic Server 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 `Diagnostic Server instance` shall call the method `ara::diag::EcuResetRequest::RequestReset` of the `ara::diag::EcuResetRequest` class instance to process an ECU Reset.] *(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.] *(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 `SoftwareClusterDiagnosticProps.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.] *(RS_Diag_04196)*

[SWS_DM_00268]{DRAFT} EcuReset positive response processing before reset [If the external processor did NOT raise an `ApApplicationError`, the `Diagnostic Server 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`, the Diagnostic Server instance shall return a positive response after the actual reset, if `ara::diag::uds_transport::UdsTransportProtocolHandler::NotifyReestablishment` is called (which could also happen after a restart of DM itself), in case the parameter `DiagnosticEcuResetClass.respondToReset` is present and set to `DiagnosticResponseToEcuResetEnum.respondAfterReset`.] (*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_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_01091]{DRAFT} Maximum time frame to trigger ExecuteReset() [The Diagnostic Server instance shall call `ara::diag::EcuResetRequest::ExecuteReset` within `p2StarServerMax` after calling `ara::diag::EcuResetRequest::RequestReset`.] (*RS_Diag_04196*)

[SWS_DM_00361]{DRAFT} EcuReset application error processing [If `ara::diag::EcuResetRequest::RequestReset` raised any of the `ara::diag::DiagErrcs` of the `ara::diag::DiagErrorDomain`, the Diagnostic Server instance shall return a negative response with the value `0x22` (`conditionsNotCorrect`).] (*RS_Diag_04196*)

[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 [In its reset preparation phase, initiated with a `ara::diag::EcuResetRequest::RequestReset` call, the `DM` shall block all incoming `UDS` requests.]([RS_Diag_04196](#))

7.5.1.8.5 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 `ClearDiagnosticInformation` 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 [7.5.2.5.5](#). It consists of resetting the `DTC` status and deleting snapshot records and extended data records.

[SWS_DM_00114]{DRAFT} Limitation to one simultaneous `DTC` clear operation
[If a `DTC` clear operation is already in progress, the `Diagnostic Server instance` shall deny an `UDS` request 0x14 and send a negative response 0x22 (`conditionsNotCorrect`).]([RS_Diag_04196](#))

[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 *DiagnosticCommonProps.clearDtcLimitation* is set to *clearAllDtc*s, 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 *DiagnosticCommonProps.clearDtcLimitation* is set to *allSupportedDtc*s, the *Diagnostic Server instance* shall allow to clear single DTCs or DTCGroups. [SWS_DM_00092] defines the possible and refused values.](RS_Diag_04117)

[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)

7.5.1.8.5.1 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 [2].

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 `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 7.5.2.5.5.

[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*)

7.5.1.8.6 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[1] and Autosar Diagnostic Extract Template [3] 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](#))

7.5.1.8.6.1 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](#))

7.5.1.8.6.2 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](#))

7.5.1.8.6.3 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](#))

7.5.1.8.6.4 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](#))

7.5.1.8.6.5 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](#))

7.5.1.8.6.6 SF 0x0A – reportSupportedDTC

[SWS_DM_00967]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x0A [If a [DiagnosticReadDTCInformation](#) of category 'REPORT_SUPPORTED_DTCS' exists, the [Diagnostic Server instance](#) shall support subfunction 0x0A (reportSupportedDTCs) 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](#))

7.5.1.8.6.7 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](#))

7.5.1.8.6.8 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](#))

7.5.1.8.6.9 SF 0x18 – reportUserDefMemoryDTCsSnapshotRecordByDTCNumber

[SWS_DM_00373]{DRAFT} Support of UDS service ReadDTCInformation, Subfunction 0x18 [The [Diagnostic Server instance](#) shall support Subfunction 0x18 (reportUserDefMemoryDTCsSnapshotRecordByDTCNumber) 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](#))

7.5.1.8.6.10 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](#))

7.5.1.8.7 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 (0x31).] ([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 7.5.4.2.1 and 7.5.4.2.2 for more details.

[SWS_DM_00571]{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 the error code.](*RS_Diag_04196*)

Note: If multiple DataIdentifier are requested within one ReadDataByIdentifier request, [SWS_DM_00571] might result in a deviation from ISO 14229-1[1] in case the AA raises an *ApApplicationError kRequestOutOfRange* (resulting in NRC 0x31). According to ISO 14229-1[1], chapter 10.2, a tester expects to receive NRC 0x31 only in case **none** of the requested DataIdentifier are supported. Handling of *ApApplicationErrors* as described in [SWS_DM_00571] might lead to NRC 0x31 on processing one of the requested DataIdentifier without checking the other requested DataIdentifier.

7.5.1.8.8 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 tester. If no data is provided by the tester, 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 similar 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_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_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_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} Blocking Timer for security access on Restart or Power down - power up cycle [The `Diagnostic Server instance` shall restart the security delay timer with the higher value of `DiagnosticCommonProps.securityDelayTimeOnBoot` / `DiagnosticSecurityLevel.securityDelayTime` of the according `DiagnosticSecurityLevel` if at least one of the stored numbers of failed attempts are greater or equal than the threshold value `DiagnosticSecurityLevel.numFailedSecurityAccess`. The behavior is equal to the behavior on runtime [SWS_DM_00272]) In case failed attempts are lower than the threshold value, the handling is equal to the behavior on runtime. (see [SWS_DM_00270] and [SWS_DM_00271])] (RS_Diag_04005)

[SWS_DM_00480]{DRAFT} Security Access Blocking Timer [If `DiagnosticSecurityAccessClass.sharedTimer` exists and is set to true, a shared delay timer instance and shared value `DiagnosticSecurityLevel.securityDelayTime` shall be used for all security levels. As long as the blocking timer is running and not expired, all requests for every `DiagnosticSecurityLevel` change with subfunction value (access type) requestSeed shall be responded with `NRC 0x37 (requiredTimeDelayNotExpired)`. (see [SWS_DM_00272]) If `DiagnosticSecurityAccessClass.sharedTimer` not exists or is set to false, an independent timer instance and timer value shall be used for each security level.](RS_Diag_04005)

[SWS_DM_CONSTR_00208]{DRAFT} Delay time value for sharedTimer [If `DiagnosticSecurityAccessClass.sharedTimer` exists and is set to true, the value

`DiagnosticSecurityLevel`. `securityDelayTime` shall be identical for all configured security levels.]([RS_Diag_04005](#))

7.5.1.8.9 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](#))

7.5.1.8.10 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 [2]. The specifications of this chapter are a sub-set of the service 0x29 specifications of ISO 14229-1:2020 [2], 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 [2].] ([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_01229]{DRAFT} Support for authentication per Diagnostic Conversation [The [Diagnostic Server instance](#) shall support the Authentication service independently for every [Diagnostic Conversation](#).] ([RS_Diag_04251](#))

NOTE : The authentication status on one [Diagnostic Conversation](#) shall not influence the access restrictions on a different Diagnostic Connection.

7.5.1.8.10.1 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_01232]{DRAFT} Handling unspecified negative return values of ara::diag::Authentication::VerifyCertificateUnidirectional method [If the method called in [SWS_DM_01230] returns any error code other than the defined error codes in [SWS_DM_01126], the *Diagnostic Server instance* shall return a Negative Response with *NRC* equal to 0x10 (generalReject).](*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.

7.5.1.8.10.2 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_01237]{DRAFT} Handling unspecified negative return values of `ara::diag::Authentication::VerifyCertificateBidirectional` method
[If the method called in [SWS_DM_01235] returns any error code other than the defined error codes in [SWS_DM_01127], the *Diagnostic Server instance* shall return a Negative Response with *NRC* equal to 0x10 (generalReject).](*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 (LPOWNSSE) 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 (POWNSSE) 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
[If 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

7.5.1.8.10.3 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_01242]{DRAFT} Handling unspecified negative return values of `ara::diag::Authentication::VerifyOwnership` method [If the method called in [SWS_DM_01240] returns any error code other than the defined error codes in [SWS_DM_01128], the *Diagnostic Server instance* shall return a Negative Response with `NRC` equal to 0x10 (generalReject).](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.

7.5.1.8.10.4 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*)

7.5.1.8.10.5 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*)

7.5.1.8.10.6 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_01250]{DRAFT} Handling unspecified negative return values of `ara::diag::Authentication::TransmitCertificate` method [If the method called in [SWS_DM_01248] returns any error code other than the defined error codes in [SWS_DM_01129], the *Diagnostic Server instance* shall return a Negative Response with *NRC* equal to 0x10 (generalReject).] (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)

7.5.1.8.11 Service 0x2A – ReadDataByPeriodicIdentifier

The processing of a *UDS* Service ReadDataByPeriodicIdentifier (0x2A) is described in ISO 14229-1[2]. 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[2].] (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[2] 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[2].](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 PID 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 PID in the list of scheduled PDIDs.](RS_Diag_04215)

[SWS_DM_01063]{DRAFT} Transmission error behavior [In case of a PID 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 PID only after the PID was transmitted successfully.](RS_Diag_04215)

7.5.1.8.12 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[2].](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 **DDID**. In default session the last defined **DDID** will be used. The client can switch to a non-default session ensure that no other testers interfere with it's **DDIDs**.

[SWS_DM_01071]{DRAFT} No persistency of defined DDIDs [If `DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling` is set to volatile, the **DM** shall initialize all **DDIDs** as not present at **DM** Initialization.]([RS_Diag_04246](#))

[SWS_DM_01072]{DRAFT} Persistency of defined DDIDs [If `DiagnosticDynamicallyDefineDataIdentifierClass.configurationHandling` is set to non-Volatile, the **DM** shall persist **DDIDs** configured by the **UDS** service 0x2C and restore the **DDID** 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 new **DID** and append the associated information received from the diagnostic request: **DID** source, position and size.]([RS_Diag_04246](#))

[SWS_DM_01074]{DRAFT} Only static DDIDs 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 an 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 DDIDs [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 `clearDynamicallyDefinedDataIdentifier` and no `dynamicallyDefinedDataIdentifier` in the request, the **DM** shall clear all the configured **DDIDs**.]([RS_Diag_04246](#))

[SWS_DM_01077]{DRAFT} Clearing individual configured DDIDs [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 0x03 `clearDynamicallyDefinedDataIdentifier` and one ore more `dynamicallyDefinedDataIdentifier` in the request, the **DM** shall clear all the **DDIDs** from the request.]([RS_Diag_04246](#))

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

[SWS_DM_01078]{DRAFT} Clear DDIDs on session change [If `DiagnosticDynamicallyDefineDataIdentifierClass.checkPerSourceId` is set to true and a diagnostic session change occurs, the **DM** shall clear all defined **DDIDs** that have `sourceDataIdentifiers` that are no longer supported in the new session and security level.]([RS_Diag_04246](#))

[SWS_DM_01079]{DRAFT} Session check for DDID [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDID can be defined in the current session according to [SWS_DM_00413]. If the DDID cannot be defined in the current session, the DM shall return an NRC 0x31 (requestOutOfRange).] (*RS_Diag_04246*)

[SWS_DM_01080]{DRAFT} Security level check for DDID [On reception of service 'DynamicallyDefineDataIdentifier' with subfunction 'defineByIdentifier', the DM shall check if the DDID can be defined in the security level according to [SWS_DM_00414]. If the DDID cannot be defined in the current security level, the DM shall return an NRC 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 the DDID 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 the DDID 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 DDID is requested with UDS service 0x22 or 0x2A.

[SWS_DM_01083]{DRAFT} Use of configured DID ports to get DDID data [If a DDID read operation is requested, the DM shall retrieve the data for the dataRecord of the DDID by using the configuration of the contained DIDs.] (*RS_Diag_04246*)

7.5.1.8.13 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](#))

7.5.1.8.14 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 subfunction 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 [The `Diagnostic Server instance` shall call the `ara::diag::GenericRoutine::Start` or `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Start` according to the mapped diagnostic interface to process the subfunction `startRoutine`.]([RS_Diag_04196](#), [RS_Diag_04224](#))

[SWS_DM_00575]{DRAFT} UDS Service RoutineControl (0x31) requestRoutineResults processing [The `Diagnostic Server instance` shall call `ara::diag::GenericRoutine::RequestResults` or `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::RequestResults` according to the mapped diagnostic interface to process the subfunction `requestRoutineResults`.]([RS_Diag_04196](#), [RS_Diag_04224](#))

[SWS_DM_00576]{DRAFT} UDS Service RoutineControl (0x31) stopRoutine processing [The `Diagnostic Server instance` shall call `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface::Stop` or `ara::diag::GenericRoutine::Stop` according to the mapped diagnostic interface to process the subfunction `stopRoutine`.]([RS_Diag_04196](#), [RS_Diag_04224](#))

7.5.1.8.15 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*)

7.5.1.8.16 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*)

7.5.1.8.17 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].

7.5.1.8.18 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].

7.5.1.8.19 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*)

7.5.1.8.20 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_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`.

7.5.1.8.21 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 7.3: 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 [2]. 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 [2].

The implementation rules mentioned in [2], 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 7.3 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.maxNumComparisionOfValueEvents` to the ISO 14229-1[1] parameter `MaxNumComparisionOfValueEvents`.] (*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*)

7.5.1.8.22 Custom Diagnostic Services

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

[SWS_DM_00502]{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 [13].

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

7.5.1.9 Cancellation of a Diagnostic Conversation

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.

[SWS_DM_00482]{DRAFT} Cancellation of a Diagnostic Conversation [Cancellation of a [Diagnostic Conversation](#) shall be performed according to [\[SWS_DM_00277\]](#), [\[SWS_DM_00278\]](#), [\[SWS_DM_00279\]](#), [\[SWS_DM_00280\]](#), [\[SWS_DM_00847\]](#).] ([RS_Diag_04167](#))

[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 on 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 on this protocol internally within the [Diagnostic Server instance](#), the [Diagnostic Server instance](#) shall abort 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 on this protocol and response transmission has not yet been started, the [Diagnostic Server instance](#) shall 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 on 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 registered notifiers.] ([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. On Cancellation of this Diagnostic Conversation `ara::diag::CancellationHandler::IsCanceled` shall return true until the `ara::diag::CancellationHandler` is destructed.](RS_Diag_04167)

7.5.2 Diagnostic Event Management

7.5.2.1 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.

7.5.2.2 Diagnostic Events

7.5.2.2.1 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](#).

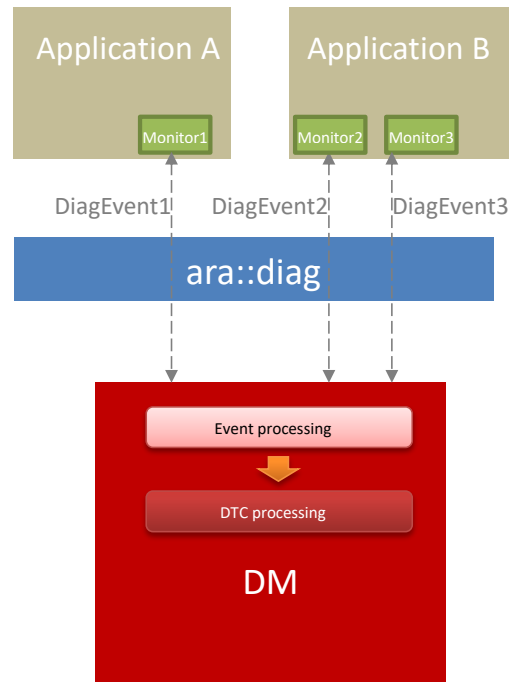


Figure 7.4: Example of diagnostic event usage

[SWS_DM_00007]{DRAFT} Uniqueness of diagnostic events [An [event](#) is unique within the system and the [DM](#) shall only support 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.] ([RS_Diag_04063](#), [RS_Diag_04179](#))

[SWS_DM_00873]{DRAFT} Diagnostic event processing interface [The [DM](#) shall provide an instance of `ara::diag::Event` per mapped `DiagnosticMonitorInterface`.] ([RS_Diag_04179](#))

The available [events](#) are derived from `DiagnosticEvent`.

7.5.2.2.2 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.

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 registered via the constructors

- `ara::diag::Monitor::Monitor`, for Monitor-internal debouncing.
- `ara::diag::Monitor::Monitor`, for counter-based debouncing.
- `ara::diag::Monitor::Monitor`, for time-based debouncing.

[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 (either

- `initMonitor` (for Monitor-internal debouncing) or
- `initMonitor` (for counter-based debouncing) or
- `initMonitor` (for time-based debouncing)

)

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 (either

- `initMonitor` (for Monitor-internal debouncing) or
- `initMonitor` (for counter-based debouncing) or
- `initMonitor` (for time-based debouncing)

)

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 (either

- `initMonitor` (for Monitor-internal debouncing) or
- `initMonitor` (for counter-based debouncing) or
- `initMonitor` (for time-based debouncing)
) with the parameter `ara::diag::InitMonitorReason` set to `kReenabled`.] ([RS_Diag_04125](#), [RS_Diag_04159](#))

[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 (either
- `initMonitor` (for Monitor-internal debouncing) or
- `initMonitor` (for counter-based debouncing) or
- `initMonitor` (for time-based debouncing)
) with the parameter `ara::diag::InitMonitorReason` set to `kDisabled`.] ([RS_Diag_04125](#), [RS_Diag_04192](#), [RS_Diag_04159](#))

7.5.2.2.3 Reporting

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.

For more details about `operation cycles` see [subsubsection 7.5.2.4](#).

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).

7.5.2.2.4 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 7.4 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 7.4: 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 7.5 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 7.5 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 7.5.2.5.2 “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	$\text{CbDTC}_{Bit0} = \text{Event}[1]_{Bit0} \mid \text{Event}[2]_{Bit0} \mid \dots \mid \text{Event}[n]_{Bit0}$
1	TestFailedThisOperationCycle	$\text{CbDTC}_{Bit1} = \text{Event}[1]_{Bit1} \mid \text{Event}[2]_{Bit1} \mid \dots \mid \text{Event}[n]_{Bit1}$
2	PendingDTC	$\text{CbDTC}_{Bit2} = \text{Event}[1]_{Bit2} \mid \text{Event}[2]_{Bit2} \mid \dots \mid \text{Event}[n]_{Bit2}$
3	ConfirmedDTC	$\text{CbDTC}_{Bit3} = \text{Event}[1]_{Bit3} \mid \text{Event}[2]_{Bit3} \mid \dots \mid \text{Event}[n]_{Bit3}$
4	TestNotCompletedSinceLastClear	$\text{CbDTC}_{Bit4} = (\text{Event}[1]_{Bit4} \mid \text{Event}[2]_{Bit4} \mid \dots \mid \text{Event}[n]_{Bit4}) \& ! \text{CbDTC}_{Bit5}$
5	TestFailedSinceLastClear	$\text{CbDTC}_{Bit5} = \text{Event}[1]_{Bit5} \mid \text{Event}[2]_{Bit5} \mid \dots \mid \text{Event}[n]_{Bit5}$
6	TestNotCompletedThisOperationCycle	$\text{CbDTC}_{Bit6} = (\text{Event}[1]_{Bit6} \mid \text{Event}[2]_{Bit6} \mid \dots \mid \text{Event}[n]_{Bit6}) \& ! \text{CbDTC}_{Bit1} \text{CbDTC}$
7	WarningIndicatorRequested	$\text{CbDTC}_{Bit7} = \text{Event}[1]_{Bit7} \mid \text{Event}[2]_{Bit7} \mid \dots \mid \text{Event}[n]_{Bit7}$

Table 7.5: Calculation of UDS status byte

In table Table 7.5 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*)

7.5.2.2.4.1 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*)

7.5.2.2.4.2 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)

7.5.2.2.5 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.

7.5.2.2.6 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 UDS DTC status bit "kTestFailed" but that a delaying threshold value must be reached before. This results in the states for `ara::diag::Event::DebouncingState`. If this threshold value is reached (FDC-equivalent is +127 (FDC_{max}) or -128 (FDC_{min})), the `ara::diag::Event::DebouncingState` is either `kFinallyDefective` or `kFinallyHealed`. This finally also leads to a change of the UDS DTC status bit "kTestFailed".

There are two kind of different debounce algorithms implemented by the DM:

- Counter-based debouncing (see 7.5.2.2.6.1 “Counter-based debouncing”))
- Time-based debouncing (see 7.5.2.2.6.2 “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 7.5.2.2.6.3 “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.

If an event is not referenced by any `DiagnosticEventToDebounceAlgorithmMapping.diagnosticEvent`, the DM does not use a debounce algorithm for this event.

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.

[SWS_DM_00874]{DRAFT} Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm [If `ara::diag::Monitor::ReportMonitorAction` is called with the parameter `action` set to `kPrepassed` or `kPrefailed` for a `diagnostic event` without assigned counter-based, time-based or monitor-internal debouncing algorithm, the DM shall trigger a Log and Trace message and ignore the reported values.] (*RS_Diag_04068*)

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 abort Monitor Constructor process (for example, via using `ara::Core::Abort`) and issues a log message for mismatching monitor.] ([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](#))

7.5.2.2.6.1 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*)

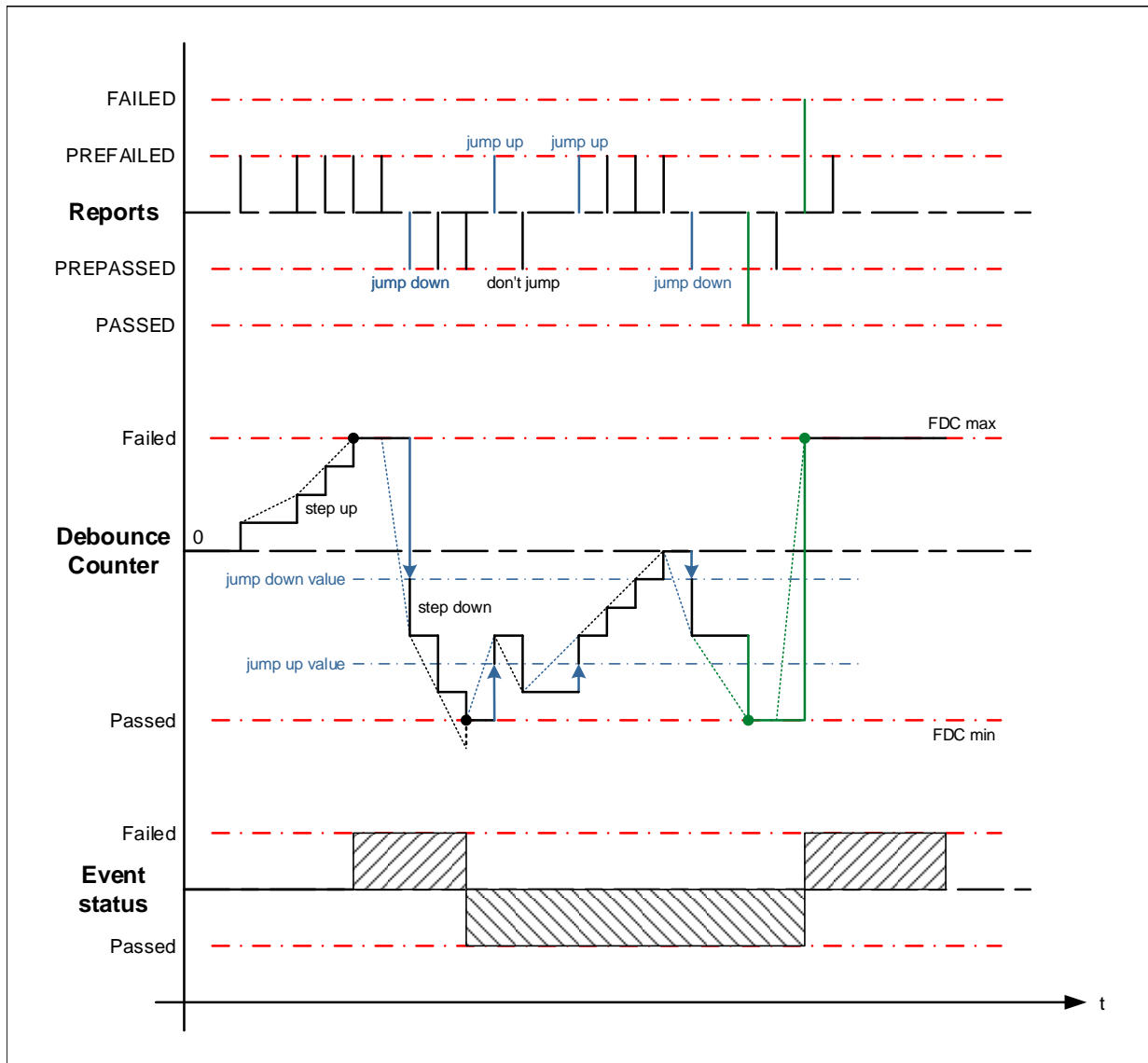


Figure 7.5: Counter-based debouncing

7.5.2.2.6.2 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. Passed or failed event states for *events* are calculated by evaluating configured thresholds of the internal debounce counter.

[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 counter init [The `DM` shall initialize the event's internal debounce counter 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 UDS, 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 internal debounce counter is used to count upon a `kPrefailed` towards the qualified failed and upon a `kPrepassed` towards a qualified passed.

[SWS_DM_00877]{DRAFT} Starting time-based event debouncing for failed [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` only when the following conditions are met:

- The debounce timer for the event is not already counting towards `kFinallyDefective`.
- The event is not already qualified as `kFinallyDefective`.

] ([RS_Diag_04225](#))

[SWS_DM_00033]{DRAFT} Debounce timer behavior upon reported failed [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 for passed [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` only when the following conditions are met:

- The debounce timer for the event is not already counting towards `kFinallyHealed`.
- The event is not already qualified as `kFinallyHealed`.

] ([RS_Diag_04225](#))

[SWS_DM_00036]{DRAFT} Debounce timer behavior upon reported passed [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](#))

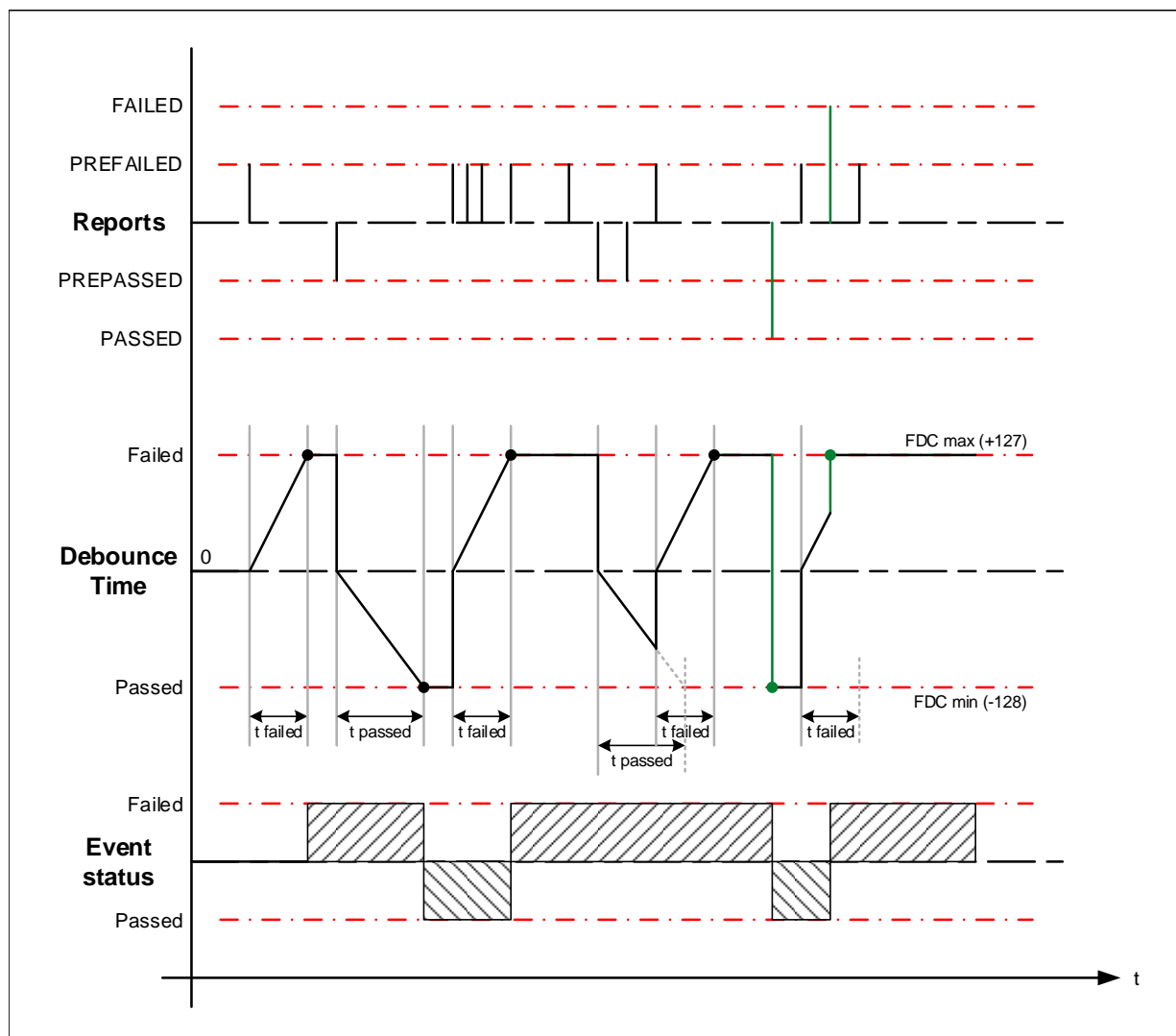


Figure 7.6: Timer based debouncing

[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`, 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*)

7.5.2.2.6.3 Monitor-internal debouncing

Monitor-internal debouncing is completely left to the diagnostic monitor `AA` which just reports the final debouncing results (`kPassed` or `kFailed`) 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].

7.5.2.2.6.4 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_00040]{DRAFT} Definition of internal debounce counter reset [To reset the internal debounce counter of an `event`, the DM shall set the corresponding internal debounce counter to zero. For time-based debouncing the debounce timer shall be stopped as well.] (*RS_Diag_04068, RS_Diag_04225*)

Only on the next call of `ara::diag::Monitor::ReportMonitorAction` with the parameter `action` set to `kPrepassed` or to `kPrefailed` the debouncing shall start over again.

[SWS_DM_00879]{DRAFT} Application resetting the internal debounce counter [If the `ara::diag::Monitor::ReportMonitorAction` method of a `ara::diag::Monitor` instance is called with the parameter `action` set to `kResetDebouncing`, the DM shall reset the internal debounce counter or timer of the corresponding event.] (*RS_Diag_04068, RS_Diag_04225*)

While resetting a timer based internal debounce counter, it is regardless if the timer is counting towards a failed or passed.

[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*)

7.5.2.2.6.5 Dependencies to enable conditions

As described in section 7.5.2.2.5 *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*)

7.5.2.2.6.6 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*)

7.5.2.2.7 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 7.5.2.2.6). 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 Clear-DiagnosticInformation 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)

7.5.2.2.8 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)

7.5.2.2.9 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 an `event` is entered in the respective `event memory`, 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.] (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.

7.5.2.3 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 [7.5.2.2.5 “EnableConditions”](#))
- `ClearDTCConditions` (see [7.5.2.5.5.2 “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](#))

7.5.2.4 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*)

7.5.2.5 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.memoryDestination`.](*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.memoryDestination`.](*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.

7.5.2.5.1 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.

7.5.2.5.1.1 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 *DiagnosticMemoryDestination.Primary.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 *DiagnosticMemoryDestination.Primary.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](#))

7.5.2.5.1.2 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.

7.5.2.5.1.3 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) ([7.5.2.5.10](#)).

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*)

7.5.2.5.2 UDS DTC Status

7.5.2.5.2.1 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 7.5.2.2.7. 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} UDS DTC status bit 'kConfirmedDTC' [If the `UDS DTC status bit` "kTestFailedThisOperationCycle" is set `DiagnosticEvent.confirmationThreshold` number of times, the `DM` shall set the `UDS`

DTC status bit 'kConfirmedDTC' to 'TRUE'.] (*RS_Diag_04136*, *RS_Diag_04180*, *RS_Diag_04157*)

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 7.5.2.5.2.3.

[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.

7.5.2.5.2.2 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*)

7.5.2.5.2.3 Indicators

Indicators can be associated with a particular *DiagnosticEvent*. Indicators or 'warning outputs' may consist of lamp(s), displayed text information or similar vendor specific expressions. There can be various *DiagnosticEvents* per indicator and one *DiagnosticEvent* can have zero, one or more different indicators assigned.

The indicators are activated and deactivated based on the configured failure and healing cycles per Event.

[SWS_DM_00221]{DRAFT} Handling indicator status [The DM shall handle the status of indicators assigned to *events* by the *DiagnosticConnectedIndicator* configuration item.] (*RS_Diag_04204*)

[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 [The `DM` shall process the "`kWarningIndicatorRequested`" bit of `DTCs` in accordance with the status vote for the assigned indicator. The "`kWarningIndicatorRequested`" bit shall be set in case the status gets confirmed and consequently the `events` shall vote positively for setting the connected indicators.]([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 the number of cycles (`DiagnosticConnectedIndicator.healingCycle`) in which the status was reported, but not failed, reaches the threshold and no user controlled WIR-bit request was set via the function `ara::diag::Event::SetLatchedWIRStatus()` for this event, the `DM` shall set the `UDS DTC status bit` "`kWarningIndicatorRequested`" to 0 and the event shall vote negatively for the activation of the indicator.]([RS_Diag_04204](#))

7.5.2.5.2.4 User controlled WarningIndicatorRequest-bit

In some cases (e.g. controlling a failsafe reaction in an application) the WIR-bit (WarningIndicatorRequest-bit) of a corresponding event in `DM` shall be set/reset by a dedicated "fail-safe `AA`".

The "failsafe `AA`" has to ensure a proper status of the WIR-bit (e.g. regarding to ISO-14229-1[2] or manufacture specific requirements).

The failsafe `AA` shall report the required WIR-status to `DM` (via the function `ara::diag::Event::SetLatchedWIRStatus` of the corresponding `ara::diag::Event` instance) and has to ensure that the current WIR-status of an event (in `DM`) fits to the current fail-safe-status in the application:

- `fail-safe reaction` active: WIR-bit shall be set to "1"
- `fail-safe reaction` not active: WIR-bit shall be set to "0"

The fail-safe `AA` has to report the status after every change of its fail-safe state.

Therefore the DM provides the function `ara::diag::Event::SetLatchedWIRStatus` to set or reset the UDS DTC status bit "kWarningIndicatorRequested" of the related DTC.

Each invocation of the function `ara::diag::Event::SetLatchedWIRStatus` of an `ara::diag::Event` instance updates the WIR-bit for the corresponding DTC

Due to not storing the Status-Bit 7 ('warningIndicatorRequested' bit) on Shutdown, the fail-safe AA has to ensure that the 'warningIndicatorRequest' bit of a DTC fits to the current failsafe status after initialization of the DM.

Setting the WIR-bit of a DTC can be controlled via `SetLatchedWIRStatus()` OR by the DM internal WIR-bit handling. (OR-Operation).

[SWS_DM_01033]{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 ControlDTC-Setting [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*)

7.5.2.5.3 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.memoryDestination` shall assign all DTCs referencing this `DiagnosticTroubleCodeProps` to the `event memory` referenced by `DiagnosticTroubleCodeProps.memoryDestination`.] (*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.memoryDestination`.] (*RS_Diag_04150, RS_Diag_04214*)

7.5.2.5.4 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 [\[SWS_DM_00895\]](#) an external processor has an error of `ara::diag::DiagnosticUdsNrcErrorDomain`, the DM shall fill the missing data with the padding value 0xFF and trigger a Log and Trace message.]([RS_Diag_04205](#))

7.5.2.5.4.1 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.

7.5.2.5.4.2 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_01084]{DRAFT} Triggering for snapshot record storage (calculated)
[If `DiagnosticMemoryDestination.typeOfFreezeFrameRecordNumeration` is set to `calculated` and the UDS DTC status bit 'testFailed' has a transition from '0' to '1', the DM shall collect and store the `snapshot record`.]([RS_Diag_04205](#))

[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 ([7.5.2.5.5](#))
- aging ([7.5.2.5.6](#))
- displacement ([7.5.2.5.10](#))

the DM doesn't trigger the notification calls for updates.

7.5.2.5.4.3 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*)

7.5.2.5.4.4 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 7.7 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 (7.5.1.8.6.4).

The [internal data elements](#) with context "DEM" in Table 7.7 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 7.5.2.5.9 "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 [7.5.2.2.9](#).

[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 [7.5.2.5.6](#).

[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*)

7.5.2.5.5 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*)

7.5.2.5.5.1 Locking of the **DTC** clearing process by a client

The **DM** supports more than one **Diagnostic Clients** as described in section 7.5.1.1.1. 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)

7.5.2.5.5.2 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 `kCondition-False`, 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)

7.5.2.5.5.3 DTC clearing triggered by application

Besides the UDS request ClearDiagnosticInformation according to section 7.5.1.8.5.1 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.

[SWS_DM_00262]{DRAFT} Common semantic behavior for ClearDTC triggered via diagnostics or application [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.](RS_Diag_04194)

[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)

7.5.2.5.6 Aging

A stored **DTC** can age in terms of reaching a 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 [The **DM** shall only support **Aging** for **DTCs**, if the corresponding configuration parameter **DiagnosticTroubleCodeProps.aging** exists.] (*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**. The **DiagnosticAging.threshold** defines the number of **Aging** cycles until **Aging**. If **DiagnosticCommonProps.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). 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_00243]{DRAFT} Aging-related UDS DTC status byte processing [As a consequence of **Aging**, 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)

[SWS_DM_00242]{DRAFT} Re-occurrence after Aging [The DM shall treat the re-occurrence of previously aged events like new events, since they were previously deleted from the event memory by `Aging`. This corresponds to all DTC-related data (i.e. counters, thresholds, etc.) being reset to their initial values.](RS_Diag_04133)

7.5.2.5.7 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)

7.5.2.5.8 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`. As the behavior for the `UDS DTC status bit 0 / 'TestFailed'` is configurable whether it is persistently stored or not (see `DiagnosticCommonProps.statusBitStorageTestFailed`), also the meaning of "active" / "passive" is influenced:

- If the 'TestFailed' bit is stored non-volatile, "**event active**" equals to 'TestFailed = 1' and "**event passive**" equals to 'TestFailed = 0'.
- If the 'TestFailed' bit is only stored volatile, additionally the information, if the event was already tested/ reported this power cycle, is required. As long, as this information is not present, the "active/passive" status is "**undefined**".

7.5.2.5.9 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](#))

7.5.2.5.10 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 `DiagnosticMemoryDestination.eventDisplacementStrategy` selects `full` and an **overflow** situation occurred in that particular **event memory**, the DM module shall perform the following selection sequence by combination of the different displacement criteria, listed by their descending importance:

- **1) Priority** (compare [SWS_DM_00916]): search through that `event memory` for entries that have the lowest priority value in that `event memory`.
- **2) Active / Passive Status:** out of the above filtered selection from 1): search for events in the following order:
 - **a.):** If the lowest priority in the `event memory` is less than the priority of the new event:
 - * **i.):** In the first place, the `DM` shall select Passive events .
 - * **ii.):** In case no Passive events are available, the `DM` shall select all events from the above filtered criteria (independent from the `UDS DTC status bits`).
 - **b.):** If the lowest priority in the `event memory` is equal to the priority of the new event:
 - * **i.):** In the first place, the `DM` shall select Passive events.
 - * **ii.):** In case no Passive events are available, the `DM` shall select events with `UDS DTC status bit "TestNotCompletedThisOperationCycle"` is set.
- **3) Oldest entry:** If the selection from the above criteria results in one or more event entries, the `DM` shall select the chronologically oldest occurred event memory entry for the displacement operation.

]([RS_Diag_04118](#), [RS_Diag_04105](#))

Details about Active / Passive Status are specified in [7.5.2.5.8 "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.

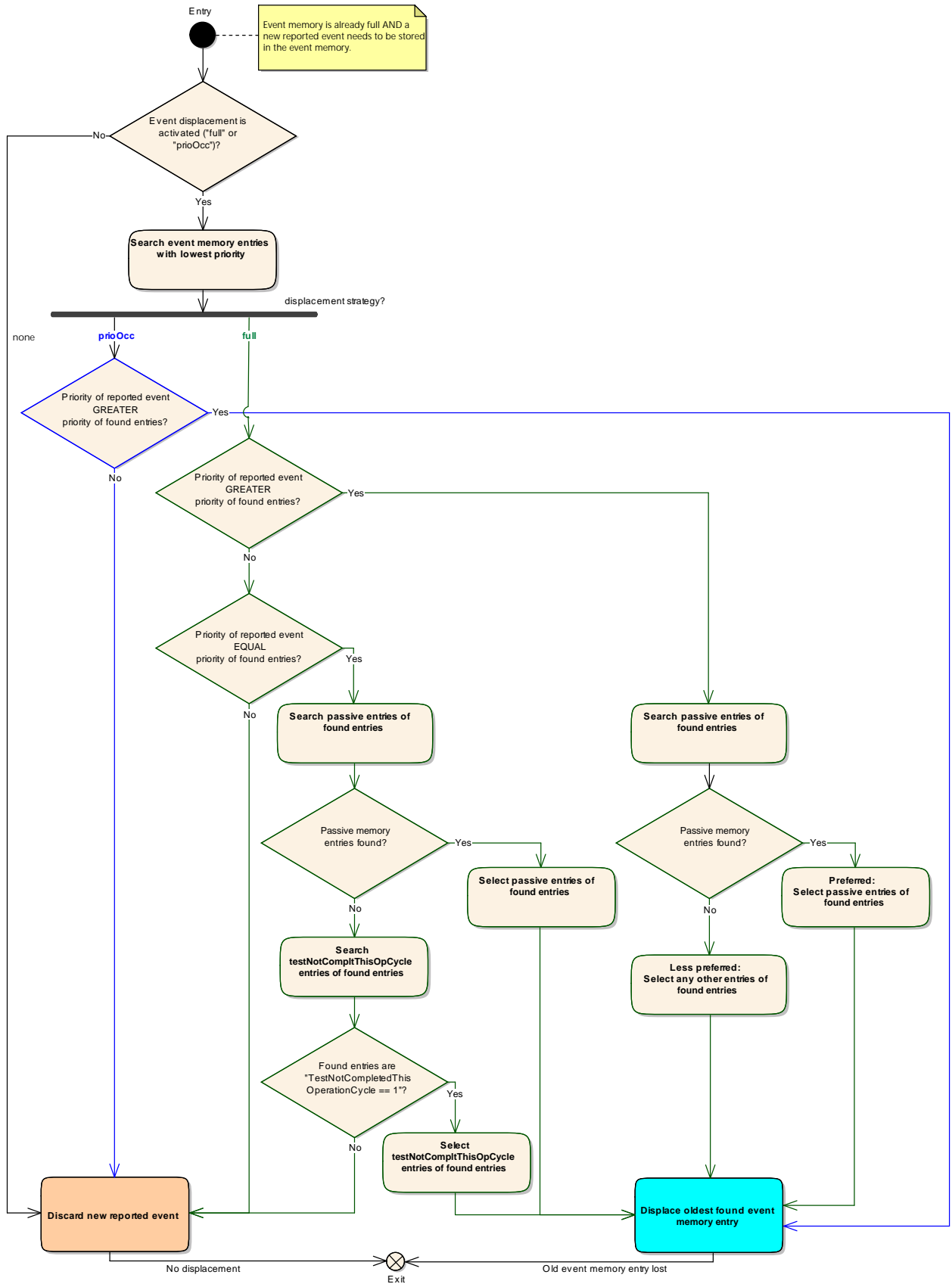


Figure 7.7: Combined displacement criteria processing

[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_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 *DiagnosticCommonProps.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*)

7.5.2.5.11 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 table 7.6 , 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*)

SSID	Name
0x02	reportDTCByStatusMask(DTCStatusMask)
0x17	reportUserDefMemoryDTCByStatusMask(DTCStatusMask, MemorySelection)

Table 7.6: 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](#))

7.5.3 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 VINDataIdentifier [A `DiagnosticDataIdentifier` with `representsVin` set to true, requires that it aggregates only one `DiagnosticParameter` which itself aggregates a `DiagnosticDataElement` having a 17 byte `uint8` array as `baseType`.] ([SRS_Eth_00026](#))

7.5.4 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`.

7.5.4.1 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 7.7 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*.

Data Provider	Content	Format	Context
DEM_AGINGCTR_DOWNCNT	Down-counting aging counter of contextual <code>DTC</code>	1 byte	DEM
DEM_AGINGCTR_UPCNT	Up-counting aging counter of contextual <code>DTC</code>	1 byte	DEM
DEM_CURRENT_FDC	Fault Detection Counter of contextual <code>DTC</code>	1 byte	DEM
DEM_CYCLES_SINCE_FIRST_FAILED	Operation Cycle Counter of contextual <code>DTC</code> – Cycles since first failed	1 byte	DEM
DEM_CYCLES_SINCE_LAST_FAILED	Operation Cycle Counter of contextual <code>DTC</code> – Cycles since last failed	1 byte	DEM
DEM_FAILED_CYCLES	Operation Cycle Counter of contextual <code>DTC</code> – Failed cycles	1 byte	DEM
DEM_MAX_FDC_DURING_CURRENT_CYCLE	Fault Detection Counter maximum value during current operation cycle of contextual <code>DTC</code>	1 byte	DEM
DEM_MAX_FDC_SINCE_LAST_CLEAR	Fault Detection Counter maximum value since last clear of contextual <code>DTC</code>	1 byte	DEM
DEM_OCCCTR	Occurrence counter of contextual <code>DTC</code>	1 byte	DEM
DEM_OVFLIND	Overflow indication of contextual <code>DTCs</code> error memory (0 = False, 1 = True)	1 byte	DEM
DEM_SIGNIFICANCE	Event significance of contextual <code>DTC</code> (refer to <code>DemDTCSignificance</code>) (0 = OCCURRENCE, 1 = FAULT)	1 byte	DEM
DEM_PRIORITY	Priority of the contextual <code>DTC</code>	1 byte	DEM
DCM_SESSION	Current session of contextual <code>Diagnostic Conversation</code>	1 byte	DCM
DCM_SECURITY_LEVEL	Current security level of contextual <code>Diagnostic Conversation</code>	1 byte	DCM
DEM_EVENT_ASSOCIATED_IDENTIFICATION	Represents the static value associated to it by <code>associatedEventIdentification</code>	4 byte	DEM

Table 7.7: 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 Table 7.7, then DM shall use the respective internally managed data value as defined in Table 7.7.](*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 7.7 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 7.7 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 7.5.4.2.1.

7.5.4.2 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 7.5.4.1,
- 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.

7.5.4.2.1 Supported Diagnostic Mappings

Details regarding the modeling of diagnostic mappings can be found in the TPS Manifest Specification [13].

7.5.4.2.2 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 at least one of the `DiagnosticDataElements` aggregated in this `DiagnosticDataIdentifier` is referenced by some `DiagnosticMapping`, then `Diagnostic Server instance` shall retrieve the data by reading data from each `DiagnosticDataElement` separately 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`. The `ara::diag::GenericUDSService::OperationOutput` is respectively composed of the requested `id` and the content of every `Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface::OperationOutput` of the related `dataElement`.](*RS_Diag_04097*)

[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.

7.5.4.2.3 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 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.

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::DiagErrorDomain`.
3. UDS NRC `ara::diag` errors: These errors can be returned by the skeletons. They are defined in the error domain `ara::diag::DiagUdsNrcErrorDomain`.

[SWS_DM_00514]{DRAFT} [

Kind:	enumeration
Symbol:	DiagErrc
Scope:	namespace <code>ara::diag</code>
Underlying type:	<code>ara::core::ErrorDomain::CodeType</code>





Syntax:	<code>enum class DiagErrc : ara::core::ErrorDomain::CodeType {...};</code>	
Values:	<code>kAlreadyOffered= 101</code>	The service is already offered.
	<code>kConfigurationMismatch= 102</code>	monitor configuration does not match dext
	<code>kDebouncingConfiguration Inconsistent= 103</code>	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
	<code>kReportIgnored= 104</code>	Enable Conditions disabled, OC not started, ...
	<code>kInvalidArgument= 105</code>	e.g. <code>kPreFailed</code> with internal debouncing
	<code>kNotOffered= 106</code>	Offer not called before reporting.
	<code>kGenericError= 107</code>	generic issue, e.g. connection to DM lost
	<code>kNoSuchDTC= 108</code>	No DTC available.
	<code>kBusy= 109</code>	Interface is busy with processing.
	<code>kFailed= 110</code>	Failed to process.
	<code>kMemoryError= 111</code>	A memory error occurred during processing.
	<code>kWrongDtc= 112</code>	A wrong DTC number was requested.
	<code>kRejected= 113</code>	Requested operation was rejected due to State Managements/machines internal state.
	<code>kResetTypeNotSupported= 114</code>	The requested Diagnostic reset type is not supported by the Diagnostic Address instance.
<code>kRequestFailed= 115</code>	Diagnostic request could not be performed successfully.	
<code>kCustomResetTypeNotSupported</code>	The requested Diagnostic custom reset type is not supported by the Diagnostic Address instance.	
Header file:	<code>#include "ara/diag/diag_error_domain.h"</code>	
Description:	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.	

|(RS_AP_00119, RS_AP_00125)

[SWS_DM_00515]{DRAFT} [

Kind:	class
Symbol:	DiagException
Scope:	namespace ara::diag
Base class:	ara::core::Exception
Syntax:	<code>class DiagException : public Exception {...};</code>
Header file:	<code>#include "ara/diag/diag_error_domain.h"</code>
Description:	Exception type thrown by Diag classes.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00516]{DRAFT} [

Kind:	function
Symbol:	DiagException(ara::core::ErrorCode err)
Scope:	class ara::diag::DiagException
Syntax:	<code>explicit DiagException (ara::core::ErrorCode err) noexcept;</code>





Parameters (in):	err	the ErrorCode
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Construct a new DiagException from an ErrorCode.	

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00517]{DRAFT} [

Kind:	class
Symbol:	DiagErrorDomain
Scope:	namespace ara::diag
Base class:	ara::core::ErrorDomain
Syntax:	<code>class DiagErrorDomain final : public ErrorDomain {...};</code>
Unique ID:	0x8000'0000'0000'0401
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Error domain for diagnostic errors.

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00518]{DRAFT} [

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

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00519]{DRAFT} [

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

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00520]{DRAFT} [

Kind:	function
Symbol:	DiagErrorDomain()
Scope:	class ara::diag::DiagErrorDomain
Syntax:	constexpr DiagErrorDomain () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Default constructor.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00521]{DRAFT} [

Kind:	function
Symbol:	Name()
Scope:	class ara::diag::DiagErrorDomain
Syntax:	const char* Name () const noexcept override;
Return value:	const char * "Diag"
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00522]{DRAFT} [

Kind:	function
Symbol:	Message(ara::core::ErrorDomain::CodeType errorCode)
Scope:	class ara::diag::DiagErrorDomain
Syntax:	const char* Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;
Parameters (in):	errorCode the error code value
Return value:	const char * the text message, never nullptr
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Translate an error code value into a text message.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00523]{DRAFT} [

Kind:	function
Symbol:	ThrowAsException(const ara::core::ErrorCode &errorCode)
Scope:	class ara::diag::DiagErrorDomain
Syntax:	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept (false) override;





Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	noexcept(false)	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Throw the exception type corresponding to the given ErrorCode.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00524]{DRAFT} [

Kind:	function	
Symbol:	GetDiagDomain()	
Scope:	namespace ara::diag	
Syntax:	constexpr const ara::core::ErrorDomain& GetDiagDomain () noexcept;	
Return value:	const ara::core::ErrorDomain &	reference to the DiagErrorDomain instance
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Obtain the reference to the single global DiagErrorDomain instance.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00525]{DRAFT} [

Kind:	function	
Symbol:	MakeErrorCode(DiagErrc code, ara::core::ErrorDomain::SupportDataType data)	
Scope:	namespace ara::diag	
Syntax:	constexpr ara::core::ErrorCode MakeErrorCode (DiagErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Create a new ErrorCode for DiagErrorDomain 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} [

Kind:	enumeration	
Symbol:	DiagOfferErrc	





Scope:	namespace ara::diag	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	enum class DiagOfferErrc : ara::core::ErrorDomain::CodeType {...};	
Values:	kAlreadyOffered= 101	The service is already offered.
	kConfigurationMismatch= 102	monitor configuration does not match dext
	kDebouncingConfiguration Inconsistent= 103	monitor debouncing configuration invalid, e.g. passed threshold larger than failed threshold...
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	The DiagOfferErrc enumeration defines the error codes for the DiagOfferErrorDomain.	

|(RS_AP_00119, RS_AP_00125)

[SWS_DM_01005]{DRAFT} [

Kind:	function	
Symbol:	MakeErrorCode(DiagOfferErrc code, ara::core::ErrorDomain::SupportDataType data)	
Scope:	namespace ara::diag	
Syntax:	constexpr ara::core::ErrorCode MakeErrorCode (DiagOfferErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Create a new ErrorCode for DiagOfferErrorDomain with the given support data type.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00991]{DRAFT} [

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

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00990]{DRAFT} [

Kind:	type alias	
Symbol:	Errc	
Scope:	class ara::diag::DiagOfferErrorDomain	





Derived from:	DiagOfferErrc
Syntax:	using Errc = DiagOfferErrc;
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Alias for the error code value enumeration.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00995]{DRAFT} [

Kind:	function	
Symbol:	ThrowAsException(const ara::core::ErrorCode &errorCode)	
Scope:	class ara::diag::DiagOfferErrorDomain	
Syntax:	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	noexcept(false)	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Throw the exception type corresponding to the given ErrorCode.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00994]{DRAFT} [

Kind:	function	
Symbol:	Message(ara::core::ErrorDomain::CodeType errorCode)	
Scope:	class ara::diag::DiagOfferErrorDomain	
Syntax:	const char* Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Translate an error code value into a text message.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00993]{DRAFT} [

Kind:	function	
Symbol:	Name()	
Scope:	class ara::diag::DiagOfferErrorDomain	
Syntax:	const char* Name () const noexcept override;	
Return value:	const char *	"DiagOffer"
Exception Safety:	noexcept	





Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.

](RS_AP_00119, RS_AP_00132)

[SWS_DM_00992]{DRAFT} [

Kind:	function
Symbol:	DiagOfferErrorDomain()
Scope:	class ara::diag::DiagOfferErrorDomain
Syntax:	constexpr DiagOfferErrorDomain () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Default constructor.

](RS_AP_00119, RS_AP_00132)

[SWS_DM_00989]{DRAFT} [

Kind:	class
Symbol:	DiagOfferErrorDomain
Scope:	namespace ara::diag
Base class:	ara::core::ErrorDomain
Syntax:	class DiagOfferErrorDomain final : public ErrorDomain {...};
Unique ID:	0x8000'0000'0000'0403
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Error domain for diagnostic offer errors.

](RS_AP_00119, RS_AP_00132)

[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} [

Kind:	enumeration				
Symbol:	DiagReportingErrc				
Scope:	namespace ara::diag				
Underlying type:	ara::core::ErrorDomain::CodeType				
Syntax:	enum class DiagReportingErrc : ara::core::ErrorDomain::CodeType {...};				
Values:	<table border="1"> <tr> <td>kAlreadyOffered= 101</td> <td>The service is already offered.</td> </tr> <tr> <td>kConfigurationMismatch= 102</td> <td>monitor configuration does not match dext</td> </tr> </table>	kAlreadyOffered= 101	The service is already offered.	kConfigurationMismatch= 102	monitor configuration does not match dext
kAlreadyOffered= 101	The service is already offered.				
kConfigurationMismatch= 102	monitor configuration does not match dext				





	kDebouncingConfiguration 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.
	kGenericError= 107	generic issue, e.g. connection to DM lost
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	The DiagReportingErrc enumeration defines the error codes for the DiagReportingError Domain. .	

]([RS_AP_00119](#), [RS_AP_00125](#))

[SWS_DM_01006]{DRAFT} [

Kind:	function	
Symbol:	MakeErrorCode(DiagReportingErrc code, ara::core::ErrorDomain::SupportDataType data)	
Scope:	namespace ara::diag	
Syntax:	constexpr ara::core::ErrorCode MakeErrorCode (DiagReportingErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;	
Parameters (in):	code	an enumeration value from future_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Create a new ErrorCode for DiagReportingErrorDomain with the given support data type.	

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00998]{DRAFT} [

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

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00997]{DRAFT} [

Kind:	type alias	
Symbol:	Errc	
Scope:	class ara::diag::DiagReportingErrorDomain	





Derived from:	DiagReportingErrc
Syntax:	using Errc = DiagReportingErrc;
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Alias for the error code value enumeration.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_01002]{DRAFT} [

Kind:	function	
Symbol:	ThrowAsException(const ara::core::ErrorCode &errorCode)	
Scope:	class ara::diag::DiagReportingErrorDomain	
Syntax:	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	noexcept(false)	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Throw the exception type corresponding to the given ErrorCode.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_01001]{DRAFT} [

Kind:	function	
Symbol:	Message(ara::core::ErrorDomain::CodeType errorCode)	
Scope:	class ara::diag::DiagReportingErrorDomain	
Syntax:	const char* Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_error_domain.h"	
Description:	Translate an error code value into a text message.	

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_01000]{DRAFT} [

Kind:	function	
Symbol:	Name()	
Scope:	class ara::diag::DiagReportingErrorDomain	
Syntax:	const char* Name () const noexcept override;	
Return value:	const char *	"DiagReporting"
Exception Safety:	noexcept	





Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.

](RS_AP_00119, RS_AP_00132)

[SWS_DM_00999]{DRAFT} [

Kind:	function
Symbol:	DiagReportingErrorDomain()
Scope:	class ara::diag::DiagReportingErrorDomain
Syntax:	constexpr DiagReportingErrorDomain () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Default constructor.

](RS_AP_00119, RS_AP_00132)

[SWS_DM_00996]{DRAFT} [

Kind:	class
Symbol:	DiagReportingErrorDomain
Scope:	namespace ara::diag
Base class:	ara::core::ErrorDomain
Syntax:	class DiagReportingErrorDomain final : public ErrorDomain {...};
Unique ID:	0x8000'0000'0000'0402
Header file:	#include "ara/diag/diag_error_domain.h"
Description:	Error domain for diagnostic reporting errors.

](RS_AP_00119, RS_AP_00132)

[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} [

Kind:	enumeration						
Symbol:	DiagUdsNrcErrc						
Scope:	namespace ara::diag						
Underlying type:	int32_t						
Syntax:	enum class DiagUdsNrcErrc : int32_t {...};						
Values:	<table border="1"> <tr> <td>kGeneralReject= 0x10</td> <td>According to ISO.</td> </tr> <tr> <td>kServiceNotSupported= 0x11</td> <td>According to ISO.</td> </tr> <tr> <td>kSubfunctionNotSupported= 0x12</td> <td>According to ISO.</td> </tr> </table>	kGeneralReject= 0x10	According to ISO.	kServiceNotSupported= 0x11	According to ISO.	kSubfunctionNotSupported= 0x12	According to ISO.
kGeneralReject= 0x10	According to ISO.						
kServiceNotSupported= 0x11	According to ISO.						
kSubfunctionNotSupported= 0x12	According to ISO.						





	kIncorrectMessageLengthOrInvalidFormat= 0x13	According to ISO.
	kResponseTooLong= 0x14	According to ISO.
	kBusyRepeatRequest= 0x21	According to ISO.
	kConditionsNotCorrect= 0x22	According to ISO.
	kRequestSequenceError= 0x24	According to ISO.
	kNoResponseFromSubnetComponent= 0x25	According to ISO.
	kFailurePreventsExecutionOfRequestedAction= 0x26	According to ISO.
	kRequestOutOfRange= 0x31	According to ISO.
	kSecurityAccessDenied= 0x33	According to ISO.
	kInvalidKey= 0x35	According to ISO.
	kExceedNumberOfAttempts= 0x36	According to ISO.
	kRequiredTimeDelayNotExpired= 0x37	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).



△

Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Description:	Specifies the types of internal errors that can occur upon calling Offer or ReportMonitorAction.

|(RS_AP_00119, RS_AP_00125)

[SWS_DM_00527]{DRAFT} [

Kind:	class
Symbol:	DiagUdsNrcException
Scope:	namespace ara::diag
Base class:	ara::core::Exception
Syntax:	<code>class DiagUdsNrcException : public Exception {...};</code>
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Description:	Exception type thrown by Diag classes.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00528]{DRAFT} [

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

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00529]{DRAFT} [

Kind:	class
Symbol:	DiagUdsNrcErrorDomain
Scope:	namespace ara::diag
Base class:	ara::core::ErrorDomain
Syntax:	<code>class DiagUdsNrcErrorDomain final : public ErrorDomain {...};</code>
Unique ID:	0x8000'0000'0000'0411
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Description:	Error domain for errors originating from several diagnostic classes.

|(RS_AP_00119, RS_AP_00132)

[SWS_DM_00530]{DRAFT} [

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

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00531]{DRAFT} [

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

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00532]{DRAFT} [

Kind:	function
Symbol:	DiagUdsNrcErrorDomain()
Scope:	class ara::diag::DiagUdsNrcErrorDomain
Syntax:	constexpr DiagUdsNrcErrorDomain () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Description:	Default constructor.

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00533]{DRAFT} [

Kind:	function
Symbol:	Name()
Scope:	class ara::diag::DiagUdsNrcErrorDomain
Syntax:	const char* Name () const noexcept override;
Return value:	const char * "DiagUdsNrc"
Exception Safety:	noexcept
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"
Description:	Return the "shortname" ApApplicationErrorDomain.SN of this error domain.

]([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00534]{DRAFT} [

Kind:	function	
Symbol:	Message(ara::core::ErrorDomain::CodeType errorCode)	
Scope:	class ara::diag::DiagUdsNrcErrorDomain	
Syntax:	const char* Message (ara::core::ErrorDomain::CodeType errorCode) const noexcept override;	
Parameters (in):	errorCode	the error code value
Return value:	const char *	the text message, never nullptr
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Description:	Translate an error code value into a text message.	

] ([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00535]{DRAFT} [

Kind:	function	
Symbol:	ThrowAsException(const ara::core::ErrorCode &errorCode)	
Scope:	class ara::diag::DiagUdsNrcErrorDomain	
Syntax:	void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;	
Parameters (in):	errorCode	the ErrorCode instance
Return value:	None	
Exception Safety:	noexcept(false)	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Description:	Throw the exception type corresponding to the given ErrorCode.	

] ([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00536]{DRAFT} [

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

] ([RS_AP_00119](#), [RS_AP_00132](#))

[SWS_DM_00537]{DRAFT} [

Kind:	function	
Symbol:	MakeErrorCode(DiagUdsNrcErrc code, ara::core::ErrorDomain::SupportDataType data)	
Scope:	namespace ara::diag	
Syntax:	constexpr ara::core::ErrorCode MakeErrorCode (DiagUdsNrcErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;	
Parameters (in):	code	an enumeration value from diag_errc
	data	a vendor-defined supplementary value
Return value:	ara::core::ErrorCode	the new ErrorCode instance
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diag_uds_nrc_error_domain.h"	
Description:	Create a new ErrorCode for DiagUdsNrcErrorDomain 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::MetaInfo class

[SWS_DM_00971]{DRAFT} [

Kind:	class
Symbol:	MetaInfo
Scope:	namespace ara::diag
Syntax:	<code>class MetaInfo final {...};</code>
Header file:	<code>#include "ara/diag/meta_info.h"</code>
Description:	Metainfo interface.

]([RS_Diag_04170](#))

8.3.1.2.2 diag::MetaInfo::MetaInfo constructor

[SWS_DM_00972]{DRAFT} [

Kind:	function
Symbol:	MetaInfo()
Scope:	class ara::diag::MetaInfo
Syntax:	MetaInfo () noexcept=delete;
Exception Safety:	noexcept
Header file:	#include "ara/diag/meta_info.h"
Description:	Constructor of MetaInfo cannot be used.

|(RS_Diag_04170)

8.3.1.2.3 diag::MetaInfo::MetaInfo copy constructor

[SWS_DM_00973]{DRAFT} [

Kind:	function
Symbol:	MetaInfo(const MetaInfo &)
Scope:	class ara::diag::MetaInfo
Syntax:	MetaInfo (const MetaInfo &)=delete;
Header file:	#include "ara/diag/meta_info.h"
Description:	Copy Constructor of MetaInfo cannot be used.

|(RS_Diag_04170)

8.3.1.2.4 diag::MetaInfo::MetaInfo move constructor

[SWS_DM_00974]{DRAFT} [

Kind:	function	
Symbol:	MetaInfo(MetaInfo &&obj)	
Scope:	class ara::diag::MetaInfo	
Syntax:	MetaInfo (MetaInfo &&obj) noexcept;	
Parameters (in):	obj	object to be moved
Exception Safety:	noexcept	
Header file:	#include "ara/diag/meta_info.h"	
Description:	Move Constructor of MetaInfo.	

|(RS_Diag_04170)

8.3.1.2.5 diag::MetaInfo::MetaInfo copy assignment operator

[SWS_DM_00975]{DRAFT} [

Kind:	function
Symbol:	operator=(const MetaInfo &)
Scope:	class ara::diag::MetaInfo
Syntax:	MetaInfo& operator= (const MetaInfo &)=delete;
Header file:	#include "ara/diag/meta_info.h"
Description:	Copy Assignment Operator of MetaInfo cannot be used.

|(RS_Diag_04170)

8.3.1.2.6 diag::MetaInfo::MetaInfo move assignment operator

[SWS_DM_00976]{DRAFT} [

Kind:	function	
Symbol:	operator=(MetaInfo &&other)	
Scope:	class ara::diag::MetaInfo	
Syntax:	MetaInfo& operator= (MetaInfo &&other) &noexcept;	
Parameters (in):	other	MetaInfo instance
Return value:	MetaInfo &	Reference to the current object
Exception Safety:	noexcept	
Header file:	#include "ara/diag/meta_info.h"	
Description:	Move Assignment Operator of MetaInfo.	

|(RS_Diag_04170)

8.3.1.2.7 diag::MetaInfo::Context enum

[SWS_DM_00977]{DRAFT} [

Kind:	enumeration	
Symbol:	Context	
Scope:	class ara::diag::MetaInfo	
Underlying type:	std::uint32_t	
Syntax:	enum class Context : std::uint32_t {...};	
Values:	kDiagnosticCommunication	service request in DCM context
	kFaultMemory	for DIDs in Snapshots
	kDoIP	for reading VIN
Header file:	#include "ara/diag/meta_info.h"	
Description:	Definition of possible call context.	

|(RS_Diag_04170)

8.3.1.2.8 diag::MetaInfo::GetValue function

[SWS_DM_00978]{DRAFT} [

Kind:	function	
Symbol:	GetValue(ara::core::StringView key)	
Scope:	class ara::diag::MetaInfo	
Syntax:	ara::core::Optional<ara::core::StringView> GetValue (ara::core::StringView key) const noexcept;	
Parameters (in):	key	identification of value to be returned
Return value:	ara::core::Optional< ara::core::StringView >	Returns value for the given key.
Exception Safety:	noexcept	
Header file:	#include "ara/diag/meta_info.h"	
Description:	Get the metainfo value for a given key.	

]([RS_Diag_04170](#))

8.3.1.2.9 diag::MetaInfo::GetContext function

[SWS_DM_00979]{DRAFT} [

Kind:	function	
Symbol:	GetContext()	
Scope:	class ara::diag::MetaInfo	
Syntax:	Context GetContext () const noexcept;	
Return value:	Context	Returns the context.
Exception Safety:	noexcept	
Header file:	#include "ara/diag/meta_info.h"	
Description:	Get the context of the invocation.	

]([RS_Diag_04170](#))

8.3.1.2.10 diag::MetaInfo::~MetaInfo destructor

[SWS_DM_00980]{DRAFT} [

Kind:	function	
Symbol:	~MetaInfo()	
Scope:	class ara::diag::MetaInfo	
Syntax:	~MetaInfo () noexcept;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/meta_info.h"	





Description:	Default destructor.
---------------------	---------------------

|(RS_Diag_04170)

Key	Context		String-Format	Description
	kDiagnostic-Communication	kFaultMemory		
kSA	X		[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	X		[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	X		"PHYS" or "FUNC"	Indicator whether request is functional or physical addressed.
kSuppPosResponse	X		"TRUE" or "FALSE"	Indicator whether the current request message contains a request to suppress the positive response message.
kSecurity-TimerElapsed	X		"TRUE" or "FALSE"	Indicates to an application, if the Security Delay Timer is elapsed.
kRequestHandle	X		[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	X		IPv4 or IPv6 address	Key for the local IP address on which the current request gets received (this might be of interest in case the ECU is multi-homed and could receive diagnostic requests via DoIP on different interfaces). The value will be either a string in IPv4 address notation (decimal representation of address parts separated with ".") or a string in IPv6 notation (hexadecimal representation of address parts separated with ":" according to section 2.2 of RFC 4291
kLocalPort	X		[0-9]{1,5}	Key for the local port number on which the current request gets received. The value will be the stringified decimal representation of the port number.
kRemoteIP	X		IPv4 or IPv6 address	Key for the remote IP address 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	X		[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		X	[0-9A-F]{6}	DTC number for which this interface is triggered

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} [

Kind:	enumeration	
Symbol:	ReentrancyType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class ReentrancyType : std::uint8_t {...};	
Values:	kFully= 0x00	The code is fully reentrant.
	kNot= 0x01	Not reentrant code.
Header file:	#include "ara/diag/reentrancy.h"	
Description:	Specifies the reentrancy types.	

]([RS_Diag_04166](#))

8.3.1.3.1 diag::DataIdentifierReentrancyType class

[SWS_DM_00936]{DRAFT} [

Kind:	struct	
Symbol:	DataIdentifierReentrancyType	
Scope:	namespace ara::diag	
Syntax:	struct DataIdentifierReentrancyType {...};	
Header file:	#include "ara/diag/reentrancy.h"	
Description:	Specifies the reentrancy type of a DataIdentifier related port.	

]([RS_Diag_04166](#))

[SWS_DM_00937]{DRAFT} [

Kind:	variable	
Symbol:	read	
Scope:	struct ara::diag::DataIdentifierReentrancyType	
Type:	ReentrancyType	
Syntax:	ReentrancyType read;	





Header file:	#include "ara/diag/reentrancy.h"
Description:	Reentrancy type for Reads.

](RS_Diag_04166)

[SWS_DM_00938]{DRAFT} [

Kind:	variable
Symbol:	write
Scope:	struct ara::diag::DataIdentifierReentrancyType
Type:	ReentrancyType
Syntax:	ReentrancyType write;
Header file:	#include "ara/diag/reentrancy.h"
Description:	Reentrancy type for Writes.

](RS_Diag_04166)

[SWS_DM_00939]{DRAFT} [

Kind:	variable
Symbol:	readWrite
Scope:	struct ara::diag::DataIdentifierReentrancyType
Type:	ReentrancyType
Syntax:	ReentrancyType readWrite;
Header file:	#include "ara/diag/reentrancy.h"
Description:	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::Monitor::CounterBased` or `ara::diag::Monitor::TimeBased`. The `specifier`, `specifier` and `specifier` are only compatible with `PortInterface` of `DiagnosticMonitorInterface`.

[SWS_DM_00542]{DRAFT} [

Kind:	class
Symbol:	Monitor
Scope:	namespace ara::diag
Syntax:	class Monitor final {...};
Header file:	#include "ara/diag/monitor.h"





Description:	Class to implement operations on diagnostic Monitor interface.
---------------------	--

]([RS_Diag_04179](#))

8.3.1.4.1 diag::Monitor::CounterBased type

[SWS_DM_00538]{DRAFT} [

Kind:	struct
Symbol:	CounterBased
Scope:	class ara::diag::Monitor
Syntax:	struct CounterBased {...};
Header file:	#include "ara/diag/monitor.h"
Description:	Represents the parameters for counter-based debouncing.

]([RS_Diag_04068](#))

[SWS_DM_00621]{DRAFT} [

Kind:	variable
Symbol:	failedThreshold
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::int16_t
Syntax:	std::int16_t failedThreshold;
Header file:	#include "ara/diag/monitor.h"
Description:	Threshold until qualified failed.

]([RS_Diag_04068](#))

[SWS_DM_00622]{DRAFT} [

Kind:	variable
Symbol:	passedThreshold
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::int16_t
Syntax:	std::int16_t passedThreshold;
Header file:	#include "ara/diag/monitor.h"
Description:	Threshold until qualified passed.

]([RS_Diag_04068](#))

[SWS_DM_00623]{DRAFT} [

Kind:	variable
Symbol:	failedStepsize
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::uint16_t
Syntax:	std::uint16_t failedStepsize;
Header file:	#include "ara/diag/monitor.h"
Description:	Stepsize per pre-failed report.

|(RS_Diag_04068)

[SWS_DM_00624]{DRAFT} [

Kind:	variable
Symbol:	passedStepsize
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::uint16_t
Syntax:	std::uint16_t passedStepsize;
Header file:	#include "ara/diag/monitor.h"
Description:	Stepsize per pre-passed report.

|(RS_Diag_04068)

[SWS_DM_00625]{DRAFT} [

Kind:	variable
Symbol:	failedJumpValue
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::int16_t
Syntax:	std::int16_t failedJumpValue;
Header file:	#include "ara/diag/monitor.h"
Description:	failed to jump value

|(RS_Diag_04068)

[SWS_DM_00626]{DRAFT} [

Kind:	variable
Symbol:	passedJumpValue
Scope:	struct ara::diag::Monitor::CounterBased
Type:	std::int16_t
Syntax:	std::int16_t passedJumpValue;
Header file:	#include "ara/diag/monitor.h"
Description:	passed to jump value

|(RS_Diag_04068)

[SWS_DM_00627]{DRAFT} [

Kind:	variable
Symbol:	useJumpToFailed
Scope:	struct ara::diag::Monitor::CounterBased
Type:	bool
Syntax:	bool useJumpToFailed;
Header file:	#include "ara/diag/monitor.h"
Description:	is jump supported

]([RS_Diag_04068](#))

[SWS_DM_00628]{DRAFT} [

Kind:	variable
Symbol:	useJumpToPassed
Scope:	struct ara::diag::Monitor::CounterBased
Type:	bool
Syntax:	bool useJumpToPassed;
Header file:	#include "ara/diag/monitor.h"
Description:	is jump supported

]([RS_Diag_04068](#))

8.3.1.4.2 diag::Monitor::TimeBased type

[SWS_DM_00539]{DRAFT} [

Kind:	struct
Symbol:	TimeBased
Scope:	class ara::diag::Monitor
Syntax:	struct TimeBased {...};
Header file:	#include "ara/diag/monitor.h"
Description:	Represents the parameters for time-based debouncing.

]([RS_Diag_04225](#))

[SWS_DM_00630]{DRAFT} [

Kind:	variable
Symbol:	passedMs
Scope:	struct ara::diag::Monitor::TimeBased
Type:	std::uint32_t
Syntax:	std::uint32_t passedMs;
Header file:	#include "ara/diag/monitor.h"





Description:	time until passed in (ms)
---------------------	---------------------------

](RS_Diag_04225)

[SWS_DM_00629]{DRAFT} [

Kind:	variable
Symbol:	failedMs
Scope:	struct ara::diag::Monitor::TimeBased
Type:	std::uint32_t
Syntax:	std::uint32_t failedMs;
Header file:	#include "ara/diag/monitor.h"
Description:	time until failed in (ms)

](RS_Diag_04225)

8.3.1.4.3 diag::InitMonitorReason enum

[SWS_DM_00540]{DRAFT} [

Kind:	enumeration	
Symbol:	InitMonitorReason	
Scope:	namespace ara::diag	
Underlying type:	std::uint32_t	
Syntax:	enum class InitMonitorReason : std::uint32_t {...};	
Values:	kClear= 0x00	Event was cleared and all internal values and states are reset.
	kRestart= 0x01	Operation cycle of the event was (re-)started.
	kReenabled= 0x02	Enable conditions are fulfilled and control DTC setting is set to on.
	kDisabled= 0x03	Enable conditions no longer fulfilled, or Control DTC setting is set to off.
Header file:	#include "ara/diag/monitor.h"	
Description:	Represents the status information reported to AAs why the monitor may be re-initialized.	

](RS_Diag_04179)

8.3.1.4.4 diag::MonitorAction enum

[SWS_DM_00541]{DRAFT} [

Kind:	enumeration	
Symbol:	MonitorAction	
Scope:	namespace ara::diag	
Underlying type:	std::uint32_t	
Syntax:	enum class MonitorAction : std::uint32_t {...};	
Values:	kPassed= 0x00	Monitor reports qualified test result passed.
	kFailed= 0x01	Monitor reports qualified test result failed.
	kPrepassed= 0x02	Monitor reports unqualified test result pre-passed.
	kPrefailed= 0x03	Monitor reports unqualified test result pre-failed.
	kFdcThresholdReached= 0x04	Monitor triggers the storage of ExtendedData Records 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.
Header file:	#include "ara/diag/monitor.h"	
Description:	Represents the status information reported by AAs being relevant for error monitoring.	

|(RS_Diag_04179)

8.3.1.4.5 diag::Monitor::Monitor constructors

[SWS_DM_00548]{DRAFT} [

Kind:	function	
Symbol:	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter)	
Scope:	class ara::diag::Monitor	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, std::function< std::int8_t()> getFaultDetectionCounter);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitorInterface
	initMonitor	Possibility to register an InitMonitor callback
	getFaultDetectionCounter	Possibility to register a function to get the current FDC for this event.
Header file:	#include "ara/diag/monitor.h"	
Description:	Monitor constructor for Monitors with Monitor-internal debouncing.	

|(RS_AP_00137, RS_Diag_04179, RS_AP_00121)

[SWS_DM_00549]{DRAFT} [

Kind:	function	
Symbol:	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues)	
Scope:	class ara::diag::Monitor	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, CounterBased defaultValues);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitorInterface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for CounterBased debouncing
Header file:	#include "ara/diag/monitor.h"	
Description:	Monitor constructor for Monitors with counter-based debouncing.	

|(RS_AP_00137, RS_Diag_04179, RS_Diag_04068, RS_AP_00121)

[SWS_DM_00550]{DRAFT} [

Kind:	function	
Symbol:	Monitor(const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, TimeBased defaultValues)	
Scope:	class ara::diag::Monitor	
Syntax:	Monitor (const ara::core::InstanceSpecifier &specifier, std::function< void(InitMonitorReason)> initMonitor, TimeBased defaultValues);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticMonitorInterface
	initMonitor	Possibility to register an InitMonitor callback
	defaultValues	Default values for TimeBased debouncing
Header file:	#include "ara/diag/monitor.h"	
Description:	Monitor constructor for Monitors with time-based debouncing.	

|(RS_AP_00137, RS_Diag_04179, RS_Diag_04225, RS_AP_00121)

8.3.1.4.6 diag::Monitor::ReportMonitorAction function

[SWS_DM_00543]{DRAFT} [

Kind:	function	
Symbol:	ReportMonitorAction(MonitorAction action)	
Scope:	class ara::diag::Monitor	
Syntax:	void ReportMonitorAction (MonitorAction action);	
Parameters (in):	action	Contains either the last (un-)qualified test result of the diagnostic monitor or commands to control the debouncing or to force a prestorage.
	Return value:	None
Header file:	#include "ara/diag/monitor.h"	
Description:	Function to report the status information being relevant for error monitoring paths.	

|(RS_Diag_04179, RS_AP_00139, RS_AP_00119)

8.3.1.4.7 diag::Monitor::Offer function

[SWS_DM_01088]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::Monitor	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/monitor.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00119, RS_AP_00139, RS_Diag_04169)

8.3.1.4.8 diag::Monitor::StopOffer function

[SWS_DM_01089]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class ara::diag::Monitor	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/monitor.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

|(RS_Diag_04169)

8.3.1.5 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} [

Kind:	class
Symbol:	GenericUDSService
Scope:	namespace ara::diag
Syntax:	<code>class GenericUDSService {...};</code>
Header file:	<code>#include "ara/diag/generic_uds_service.h"</code>
Description:	Generic UDS interface.

]([RS_Diag_04169](#))

8.3.1.5.1 diag::GenericUDSService::OperationOutput type

[SWS_DM_00578]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class ara::diag::GenericUDSService
Syntax:	<code>struct OperationOutput {...};</code>
Header file:	<code>#include "ara/diag/generic_uds_service.h"</code>
Description:	Response data of positive response message.

]([RS_Diag_04169](#), [RS_Diag_04172](#))

8.3.1.5.2 diag::GenericUDSService::OperationOutput::responseData

[SWS_DM_00632]{DRAFT} [

Kind:	variable
Symbol:	responseData
Scope:	struct ara::diag::GenericUDSService::OperationOutput
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector<std::uint8_t> responseData;</code>
Header file:	<code>#include "ara/diag/generic_uds_service.h"</code>
Description:	Content of positive response message (without SID)

]([RS_Diag_04169](#), [RS_Diag_04172](#))

8.3.1.5.3 diag::GenericUDSService::GenericUDSService function

[SWS_DM_00616]{DRAFT} [

Kind:	function	
Symbol:	GenericUDSService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::GenericUDSService	
Syntax:	explicit GenericUDSService (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/generic_uds_service.h"	
Description:	Constructor of GenericUDSService.	

](RS_AP_00137, RS_Diag_04169)

8.3.1.5.4 diag::GenericUDSService::~~GenericUDSService function

[SWS_DM_00584]{DRAFT} [

Kind:	function	
Symbol:	~GenericUDSService()	
Scope:	class ara::diag::GenericUDSService	
Syntax:	virtual ~GenericUDSService () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/generic_uds_service.h"	
Description:	Destructor of GenericUDSService.	

](RS_AP_00134, RS_Diag_04169)

8.3.1.5.5 diag::GenericUDSService::Offer function

[SWS_DM_00619]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::GenericUDSService	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/generic_uds_service.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

](RS_AP_00139, RS_AP_00119, RS_Diag_04169)

8.3.1.5.6 diag::GenericUDSService::StopOffer function

[SWS_DM_00620]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::GenericUDSService
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/generic_uds_service.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

]([RS_Diag_04169](#))

8.3.1.5.7 diag::GenericUDSService::HandleMessage function

[SWS_DM_00618]{DRAFT} [

Kind:	function	
Symbol:	HandleMessage(std::uint8_t sid, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::GenericUDSService	
Syntax:	virtual ara::core::Future<OperationOutput> HandleMessage (std::uint8_t sid, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	sid	Diagnostic Request Service Identifier.
	requestData	Diagnostic request data (starting after SID).
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< OperationOutput >	a Result with either a OperationOutput or an error
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResourceTemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
Header file:	#include "ara/diag/generic_uds_service.h"	
Description:	Called for any request message.	

|(RS_AP_00119, RS_AP_00138, RS_Diag_04169, RS_Diag_04170, RS_Diag_04172, RS_Diag_04173)

8.3.1.6 GenericDataIdentifier class

This interface allows a generic implementation of an data identifier handler. Multiple [DiagnosticServiceGenericMappings](#) can reference to the same [PortPrototype](#). The `specifier` is only compatible with [PortInterface](#) of [DiagnosticDataIdentifierGenericInterface](#).

[SWS_DM_00607]{DRAFT} [

Kind:	class
Symbol:	GenericDataIdentifier
Scope:	namespace ara::diag
Syntax:	<code>class GenericDataIdentifier {...};</code>
Header file:	#include "ara/diag/generic_data_identifier.h"
Description:	Generic DataIdentifier interface.

|(RS_Diag_04169)

8.3.1.6.1 diag::GenericDataIdentifier::OperationOutput type

[SWS_DM_00641]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class ara::diag::GenericDataIdentifier
Syntax:	struct OperationOutput {...};
Header file:	#include "ara/diag/generic_data_identifier.h"
Description:	Response data of positive response message.

|(RS_Diag_04169, RS_Diag_04172)

[SWS_DM_00631]{DRAFT} [

Kind:	variable
Symbol:	responseData
Scope:	struct ara::diag::GenericDataIdentifier::OperationOutput
Type:	ara::core::Vector< std::uint8_t >
Syntax:	ara::core::Vector<std::uint8_t> responseData;
Header file:	#include "ara/diag/generic_data_identifier.h"
Description:	Content of positive response message (without DataIdentifier)

|(RS_AP_00137, RS_Diag_04169)

8.3.1.6.2 diag::GenericDataIdentifier::GenericDataIdentifier function

[SWS_DM_00634]{DRAFT} [

Kind:	function	
Symbol:	GenericDataIdentifier(const ara::core::InstanceSpecifier &specifier, DataIdentifierReentrancyType reentrancyType)	
Scope:	class ara::diag::GenericDataIdentifier	
Syntax:	explicit GenericDataIdentifier (const ara::core::InstanceSpecifier &specifier, DataIdentifierReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDataIdentifierGenericInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant for reads, writes or both
Header file:	#include "ara/diag/generic_data_identifier.h"	
Description:	Class for an GenericDataIdentifier.	

|(RS_AP_00137, RS_Diag_04169)

8.3.1.6.3 diag::GenericDataIdentifier::~GenericDataIdentifier function

[SWS_DM_00635]{DRAFT} [

Kind:	function
Symbol:	~GenericDataIdentifier()
Scope:	class ara::diag::GenericDataIdentifier
Syntax:	virtual ~GenericDataIdentifier () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/generic_data_identifier.h"
Description:	Destructor of class GenericDataIdentifier.

|(RS_AP_00134, RS_Diag_04169)

8.3.1.6.4 diag::GenericDataIdentifier::Offer function

[SWS_DM_00638]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::GenericDataIdentifier	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void > –	
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/generic_data_identifier.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00139, RS_AP_00119, RS_Diag_04169)

8.3.1.6.5 diag::GenericDataIdentifier::StopOffer function

[SWS_DM_00639]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::GenericDataIdentifier
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/generic_data_identifier.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

|(RS_Diag_04169)

8.3.1.6.6 diag::GenericDataIdentifier::Read function

[SWS_DM_00636]{DRAFT} [

Kind:	function	
Symbol:	Read(std::uint16_t dataIdentifier, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::GenericDataIdentifier	
Syntax:	virtual ara::core::Future<OperationOutput> Read (std::uint16_t dataIdentifier, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	dataIdentifier	the corresponding DataIdentifier
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.	



△

	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResourceTemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
Header file:	#include "ara/diag/generic_data_identifier.h"	
Description:	Called for ReadDataByIdentifier request for this DiagnosticDataIdentifier.	

|(RS_AP_00138, RS_AP_00119, RS_Diag_04169, RS_Diag_04170, RS_Diag_04172)

8.3.1.6.7 diag::GenericDataIdentifier::Write function

[SWS_DM_00637]{DRAFT} [

Kind:	function	
Symbol:	Write(std::uint16_t dataIdentifier, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::GenericDataIdentifier	
Syntax:	virtual ara::core::Future<void> Write (std::uint16_t dataIdentifier, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	dataIdentifier	the corresponding DataIdentifier
	requestData	Content of request message (without DataIdentifier)
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.	



△

	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/generic_data_identifier.h"	
Description:	Called for WriteDataByIdentifier request for this DiagnosticDataIdentifier.	

|(RS_AP_00138, RS_AP_00119, RS_Diag_04169, RS_Diag_04170)

8.3.1.7 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} [

Kind:	class
Symbol:	GenericRoutine
Scope:	namespace ara::diag
Syntax:	<code>class GenericRoutine {...};</code>
Header file:	<code>#include "ara/diag/generic_routine.h"</code>
Description:	Generic Routine interface.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.1.7.1 diag::GenericRoutine::OperationOutput type

[SWS_DM_00551]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class ara::diag::GenericRoutine
Syntax:	<code>struct OperationOutput {...};</code>
Header file:	<code>#include "ara/diag/generic_routine.h"</code>
Description:	Response data of positive response message.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

[SWS_DM_00633]{DRAFT} [

Kind:	variable
Symbol:	responseData
Scope:	struct ara::diag::GenericRoutine::OperationOutput
Type:	<code>ara::core::Vector< std::uint8_t ></code>
Syntax:	<code>ara::core::Vector< std::uint8_t > responseData;</code>
Header file:	<code>#include "ara/diag/generic_routine.h"</code>
Description:	Content of positive response message (without RoutineIdentifier)

]([RS_AP_00137](#), [RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.1.7.2 diag::GenericRoutine::GenericRoutine function

[SWS_DM_00552]{DRAFT} [

Kind:	function	
Symbol:	GenericRoutine(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::GenericRoutine	
Syntax:	explicit GenericRoutine (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticRoutineGenericInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/generic_routine.h"	
Description:	Class for an GenericRoutine.	

]([RS_AP_00137](#), [RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.1.7.3 diag::GenericRoutine::~~GenericRoutine function

[SWS_DM_00553]{DRAFT} [

Kind:	function	
Symbol:	~GenericRoutine()	
Scope:	class ara::diag::GenericRoutine	
Syntax:	virtual ~GenericRoutine () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/generic_routine.h"	
Description:	Destructor of class GenericRoutine.	

]([RS_AP_00134](#), [RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.1.7.4 diag::GenericRoutine::Offer function

[SWS_DM_00557]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::GenericRoutine	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.





Header file:	#include "ara/diag/generic_routine.h"
Description:	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](#))

8.3.1.7.5 diag::GenericRoutine::StopOffer function

[SWS_DM_00558]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::GenericRoutine
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/generic_routine.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.1.7.6 diag::GenericRoutine::Start function

[SWS_DM_00554]{DRAFT} [

Kind:	function	
Symbol:	Start(std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::GenericRoutine	
Syntax:	virtual ara::core::Future<OperationOutput> Start (std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without Routine Identifier)
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/generic_routine.h"	
Description:	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](#))

8.3.1.7.7 diag::GenericRoutine::Stop function

[SWS_DM_00555]{DRAFT} [

Kind:	function	
Symbol:	Stop(std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::GenericRoutine	
Syntax:	virtual ara::core::Future<OperationOutput> Stop (std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	routineId	the corresponding RoutineIdentifier
	requestData	Content of request message (without Routine Identifier)
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled





Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.	
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.	
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.	





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/generic_routine.h"	
Description:	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)

8.3.1.7.8 diag::GenericRoutine::RequestResults function

[SWS_DM_00556]{DRAFT} [

Kind:	function
Symbol:	RequestResults(std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)
Scope:	class ara::diag::GenericRoutine
Syntax:	virtual ara::core::Future<OperationOutput> RequestResults (std::uint16_t routineId, ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;





Parameters (in):	routineld	the corresponding RoutineIdentifier
	requestData	Content of request message (without Routine Identifier)
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.



△

	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/generic_routine.h"	
Description:	Called for RoutineControl with SubFunction RequestResults request for this DiagnosticRoutine Identifier.	

|(RS_AP_00119, RS_Diag_04169, RS_Diag_04170, RS_Diag_04224)

8.3.1.8 CancellationHandler class

[SWS_DM_00608]{DRAFT} [

Kind:	class
Symbol:	CancellationHandler
Scope:	namespace ara::diag
Syntax:	<code>class CancellationHandler final {...};</code>
Header file:	<code>#include "ara/diag/cancellation_handler.h"</code>
Description:	CancellationHandler contains a shared state if the processing should be canceled .

]([RS_Diag_04169](#))

8.3.1.8.1 diag::CancellationHandler::CancellationHandler function

[SWS_DM_00609]{DRAFT} [

Kind:	function
Symbol:	CancellationHandler()
Scope:	class ara::diag::CancellationHandler
Syntax:	<code>CancellationHandler ()=delete;</code>
Header file:	<code>#include "ara/diag/cancellation_handler.h"</code>
Description:	Constructor of CancellationHandler cannot be used.

]([RS_Diag_04169](#))

[SWS_DM_00610]{DRAFT} [

Kind:	function
Symbol:	CancellationHandler(CancellationHandler &&)
Scope:	class ara::diag::CancellationHandler
Syntax:	<code>CancellationHandler (CancellationHandler &&) noexcept=default;</code>
DIRECTION NOT DEFINED	CancellationHandler && -
Exception Safety:	noexcept
Header file:	<code>#include "ara/diag/cancellation_handler.h"</code>
Description:	Move constructor of CancellationHandler.

]([RS_AP_00133](#), [RS_Diag_04169](#))

[SWS_DM_00611]{DRAFT} [

Kind:	function
Symbol:	CancellationHandler(CancellationHandler &)
Scope:	class ara::diag::CancellationHandler
Syntax:	<code>CancellationHandler (CancellationHandler &)=delete;</code>
Header file:	<code>#include "ara/diag/cancellation_handler.h"</code>





Description:	Copy constructor of CancellationHandler cannot be used.
---------------------	---

](RS_Diag_04169)

[SWS_DM_00612]{DRAFT} [

Kind:	function	
Symbol:	operator=(CancellationHandler &&)	
Scope:	class ara::diag::CancellationHandler	
Syntax:	CancellationHandler& operator= (CancellationHandler &&) noexcept=default;	
DIRECTION NOT DEFINED	CancellationHandler &&	–
Return value:	CancellationHandler &	–
Exception Safety:	noexcept	
Header file:	#include "ara/diag/cancellation_handler.h"	
Description:	Move assignment operator of CancellationHandler.	

](RS_AP_00133, RS_Diag_04169)

[SWS_DM_00613]{DRAFT} [

Kind:	function	
Symbol:	operator=(CancellationHandler &)	
Scope:	class ara::diag::CancellationHandler	
Syntax:	CancellationHandler& operator= (CancellationHandler &)=delete;	
Header file:	#include "ara/diag/cancellation_handler.h"	
Description:	Copy assignment operator of CancellationHandler cannot be used.	

](RS_Diag_04169)

8.3.1.8.2 diag::CancellationHandler::IsCanceled function

[SWS_DM_00614]{DRAFT} [

Kind:	function	
Symbol:	IsCanceled()	
Scope:	class ara::diag::CancellationHandler	
Syntax:	bool IsCanceled () const;	
Return value:	bool	–
Header file:	#include "ara/diag/cancellation_handler.h"	
Description:	Returns true in if the diagnostic service execution is cancelled in DM.	

](RS_Diag_04169)

8.3.1.8.3 diag::CancellationHandler::SetNotifier function

[SWS_DM_00615]{DRAFT} [

Kind:	function	
Symbol:	SetNotifier(std::function< void()> notifier)	
Scope:	class ara::diag::CancellationHandler	
Syntax:	void SetNotifier (std::function< void()> notifier);	
DIRECTION NOT DEFINED	notifier	-
Return value:	None	
Header file:	#include "ara/diag/cancellation_handler.h"	
Description:	Registering a notifier function which is called if the diagnostic service execution is canceled in DM.	

]([RS_Diag_04169](#))

8.3.1.9 Event class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticEventInterface](#).

[SWS_DM_00646]{DRAFT} [

Kind:	class	
Symbol:	Event	
Scope:	namespace ara::diag	
Syntax:	class Event {...};	
Header file:	#include "ara/diag/event.h"	
Description:	Class to implement operations on diagnostic Events.	

]([RS_Diag_04151](#))

8.3.1.9.1 diag::DTCFormatType type

[SWS_DM_00642]{DRAFT} [

Kind:	enumeration	
Symbol:	DTCFormatType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class DTCFormatType : std::uint8_t {...};	
Values:	kDTCFormatOBD= 0	SAE_J2012-DA_DTCFormat_00 as defined in ISO 15031-6 specification.



△

	kDTCFormatUDS= 1	ISO_14229-1_DTCFormat as defined in ISO 14229-1 specification.
	kDTCFormatJ1939= 2	SAE_J1939-73_DTCFormat as defined in SAE J1939-73.
Header file:	#include "ara/diag/event.h"	
Description:	Represents the type of the DTC format according to ISO 14229-1.	

]([RS_Diag_04201](#), [RS_AP_00125](#))

8.3.1.9.2 diag::EventStatusBit type

[SWS_DM_00643]{DRAFT} [

Kind:	enumeration	
Symbol:	EventStatusBit	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class EventStatusBit : std::uint8_t {...};	
Values:	kTestFailed	bit 0: TestFailed
	kTestFailedThisOperationCycle	bit 1: TestFailedThisOperationCycle
	kTestNotCompletedThisOperationCycle	bit 6: TestNotCompletedThisOperationCycle
Header file:	#include "ara/diag/event.h"	
Description:	Single event status bits.	

]([RS_Diag_04151](#), [RS_AP_00125](#))

8.3.1.9.3 diag::Event::EventStatusByte type

[SWS_DM_00644]{DRAFT} [

Kind:	struct
Symbol:	EventStatusByte
Scope:	class ara::diag::Event
Syntax:	struct EventStatusByte {...};
Header file:	#include "ara/diag/event.h"
Description:	Current event status byte, bit-encoded.

]([RS_Diag_04151](#))

8.3.1.9.4 diag::Event::DebouncingState type

[SWS_DM_00645]{DRAFT} [

Kind:	enumeration	
Symbol:	DebouncingState	
Scope:	class ara::diag::Event	
Underlying type:	std::uint8_t	
Syntax:	enum class DebouncingState : std::uint8_t {...};	
Values:	kNeutral= 0x00	Neutral (corresponds to FDC = 0)
	kTemporarilyDefective= 0x01	Temporarily Defective (corresponds to 0 < FDC < 127)
	kFinallyDefective= 0x02	finally Defective (corresponds to FDC = 127)
	kTemporarilyHealed= 0x04	temporarily healed (corresponds to -128 < FDC < 0)
	kFinallyHealed= 0x08	finally healed (corresponds to FDC = -128)
Header file:	#include "ara/diag/event.h"	
Description:	Debounce status of event .	

]([RS_Diag_04068](#), [RS_Diag_04225](#), [RS_AP_00125](#))

8.3.1.9.5 diag::Event::Event constructor

[SWS_DM_00647]{DRAFT} [

Kind:	function	
Symbol:	Event(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::Event	
Syntax:	explicit Event (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticEventInterface
Header file:	#include "ara/diag/event.h"	
Description:	Constructor fct. for objects of class Event.	

]([RS_Diag_04151](#), [RS_AP_00137](#))

8.3.1.9.6 diag::Event::~~Event destructor

[SWS_DM_00648]{DRAFT} [

Kind:	function	
Symbol:	~Event()	
Scope:	class ara::diag::Event	
Syntax:	~Event () noexcept=default;	



△

Exception Safety:	noexcept
Header file:	#include "ara/diag/event.h"
Description:	Destructor of class Event.

]([RS_Diag_04151](#), [RS_AP_00134](#))

8.3.1.9.7 diag::Event::GetEventStatus function

[SWS_DM_00649]{DRAFT} [

Kind:	function	
Symbol:	GetEventStatus()	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<EventStatusByte> GetEventStatus ();	
Return value:	ara::core::Result< EventStatusByte >	the current diagnostic event status
Header file:	#include "ara/diag/event.h"	
Description:	Returns the current diagnostic event status.	

]([RS_Diag_04151](#), [RS_AP_00139](#))

8.3.1.9.8 diag::Event::SetEventStatusChangedNotifier function

[SWS_DM_00650]{DRAFT} [

Kind:	function	
Symbol:	SetEventStatusChangedNotifier(std::function< void(EventStatusByte)> notifier)	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<void> SetEventStatusChangedNotifier (std::function< void(EventStatusByte)> notifier);	
Parameters (in):	notifier	The function to be called if a diagnostic event is changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kInvalid Argument	given argument is invalid (pointer).
Header file:	#include "ara/diag/event.h"	
Description:	Register a notifier function which is called if a diagnostic event is changed.	

]([RS_Diag_04183](#), [RS_AP_00139](#))

8.3.1.9.9 diag::Event::GetLatchedWIRStatus function

[SWS_DM_00651]{DRAFT} [

Kind:	function	
Symbol:	GetLatchedWIRStatus()	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<bool> GetLatchedWIRStatus ();	
Return value:	ara::core::Result< bool >	the current warning indicator status
Header file:	#include "ara/diag/event.h"	
Description:	Returns the current warning indicator status.	

|(RS_Diag_04204, RS_AP_00139)

8.3.1.9.10 diag::Event::SetLatchedWIRStatus function

[SWS_DM_00652]{DRAFT} [

Kind:	function	
Symbol:	SetLatchedWIRStatus(bool status)	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<void> SetLatchedWIRStatus (bool status);	
Parameters (in):	status	Limp-home status as determined by the AA. '0' means limp-home not active; '1' means limp-home active;
Return value:	ara::core::Result< void >	-
Header file:	#include "ara/diag/event.h"	
Description:	Set the warning indicator status.	

|(RS_Diag_04151, RS_AP_00139)

8.3.1.9.11 diag::Event::GetDTCNumber function

[SWS_DM_00653]{DRAFT} [

Kind:	function	
Symbol:	GetDTCNumber(DTCFormatType dtcFormat)	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<std::uint32_t> GetDTCNumber (DTCFormatType dtcFormat);	
Parameters (in):	dtcFormat	Define DTC format for the return value.
Return value:	ara::core::Result< std::uint32_t >	DTC number in respective DTCFormatType
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
Header file:	#include "ara/diag/event.h"	
Description:	Returns the DTC-ID related to this event instance.	

|(RS_Diag_04201, RS_AP_00139, RS_AP_00119)

8.3.1.9.12 diag::Event::GetDebouncingStatus function

[SWS_DM_00654]{DRAFT} [

Kind:	function	
Symbol:	GetDebouncingStatus()	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<DebouncingState> GetDebouncingStatus ();	
Return value:	ara::core::Result< DebouncingState >	Return the current debouncing state of this event.
Header file:	#include "ara/diag/event.h"	
Description:	Get the current debouncing status .	

]([RS_Diag_04068](#), [RS_Diag_04225](#), [RS_AP_00139](#))

8.3.1.9.13 diag::Event::GetTestComplete function

[SWS_DM_00655]{DRAFT} [

Kind:	function	
Symbol:	GetTestComplete()	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<bool> GetTestComplete ();	
Return value:	ara::core::Result< bool >	Return the current test_completed-state of this event. "true", if FDC = -128 or FDC = 127; "false" in all other cases.
Header file:	#include "ara/diag/event.h"	
Description:	Get the status if the event has matured to test completed (corresponds to FDC = -128 or FDC = 127).	

]([RS_Diag_04151](#), [RS_AP_00139](#))

8.3.1.9.14 diag::Event::GetFaultDetectionCounter function

[SWS_DM_00656]{DRAFT} [

Kind:	function	
Symbol:	GetFaultDetectionCounter()	
Scope:	class ara::diag::Event	
Syntax:	ara::core::Result<std::int8_t> GetFaultDetectionCounter ();	
Return value:	ara::core::Result< std::int8_t >	current FaultDetectionCounter value.
Header file:	#include "ara/diag/event.h"	
Description:	Returns the current value of Fault Detection Counter of this event.	

]([RS_Diag_04068](#), [RS_AP_00139](#))

8.3.1.10 DTCInformation class

The *InstanceSpecifier* is only compatible with `PortInterface` of `DiagnosticDTCInformationInterface`.

[SWS_DM_00657]{DRAFT} [

Kind:	class
Symbol:	DTCInformation
Scope:	namespace ara::diag
Syntax:	<code>class DTCInformation {...};</code>
Header file:	<code>#include "ara/diag/dtc_information.h"</code>
Description:	Class to implement operations on DTC informations per configured DiagnosticMemory Destination.

]([RS_Diag_04150](#), [RS_Diag_04105](#))

8.3.1.10.1 diag::ControlDtcStatusType type

[SWS_DM_00663]{DRAFT} [

Kind:	enumeration	
Symbol:	ControlDtcStatusType	
Scope:	namespace ara::diag	
Underlying type:	uint8_t	
Syntax:	<code>enum class ControlDtcStatusType : uint8_t {...};</code>	
Values:	kDTCSettingOn= 0x00	Updating of diagnostic trouble code status bits is under normal operating conditions.
	kDTCSettingOff= 0x01	Updating of diagnostic trouble code status bits is stopped.
Header file:	<code>#include "ara/diag/dtc_information.h"</code>	
Description:	Type for ControlDTCStatus status as requested by UDS service 0x85 ControlDTCSetting.	

]([RS_Diag_04159](#))

8.3.1.10.2 diag::UdsDtcStatusBitType type

[SWS_DM_00658]{DRAFT} [

Kind:	enumeration
Symbol:	UdsDtcStatusBitType
Scope:	namespace ara::diag
Underlying type:	std::uint8_t
Syntax:	<code>enum class UdsDtcStatusBitType : std::uint8_t {...};</code>





Values:	kTestFailed= 0x01	bit 0: TestFailed
	kTestFailedThisOperationCycle= 0x02	bit 1: TestFailedThisOperationCycle
	kPendingDTC= 0x04	bit 2: PendingDTC
	kConfirmedDTC= 0x08	bit 3: ConfirmedDTC
	kTestNotCompletedSinceLastClear= 0x10	bit 4: TestNotCompletedSinceLastClear
	kTestFailedSinceLastClear= 0x20	bit 5: TestFailedSinceLastClear
	kTestNotCompletedThisOperationCycle= 0x40	bit 6: TestNotCompletedThisOperationCycle
	kWarningIndicatorRequested= 0x80	bit 7: WarningIndicatorRequested
Header file:	#include "ara/diag/dtc_information.h"	
Description:	UDS DTC status bits according to ISO 14229-1.	

]([RS_Diag_04067](#), [RS_Diag_04151](#))

8.3.1.10.3 diag::DTCInformation::UdsDtcStatusByteType type

[SWS_DM_00659]{DRAFT} [

Kind:	struct
Symbol:	UdsDtcStatusByteType
Scope:	class ara::diag::DTCInformation
Syntax:	struct UdsDtcStatusByteType {...};
Header file:	#include "ara/diag/dtc_information.h"
Description:	Type for UDS DTC status byte.

]([RS_Diag_04067](#), [RS_Diag_04151](#))

8.3.1.10.4 diag::DTCInformation::SnapshotDataIdentifierType type

[SWS_DM_00660]{DRAFT} [

Kind:	struct
Symbol:	SnapshotDataIdentifierType
Scope:	class ara::diag::DTCInformation
Syntax:	struct SnapshotDataIdentifierType {...};
Header file:	#include "ara/diag/dtc_information.h"
Description:	Type for SnapshotDataIdentifierType status.

]([RS_Diag_04205](#))

8.3.1.10.5 diag::DTCInformation::SnapshotDataRecordType type

[SWS_DM_00661]{DRAFT} [

Kind:	struct
Symbol:	SnapshotDataRecordType
Scope:	class ara::diag::DTCInformation
Syntax:	struct SnapshotDataRecordType {...};
Header file:	#include "ara/diag/dtc_information.h"
Description:	Type for SnapshotDataRecordType status.

]([RS_Diag_04205](#))

8.3.1.10.6 diag::DTCInformation::SnapshotRecordUpdatedType type

[SWS_DM_00662]{DRAFT} [

Kind:	struct
Symbol:	SnapshotRecordUpdatedType
Scope:	class ara::diag::DTCInformation
Syntax:	struct SnapshotRecordUpdatedType {...};
Header file:	#include "ara/diag/dtc_information.h"
Description:	Type for SnapshotRecordUpdatedType status.

]([RS_Diag_04205](#))

8.3.1.10.7 diag::DTCInformation::DTCInformation function

[SWS_DM_00664]{DRAFT} [

Kind:	function	
Symbol:	DTCInformation(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	explicit DTCInformation (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDTCInformationInterface
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Constructor for a DTCInformation instance which allows for DTC related operation per DiagnosticMemoryDestination.	

]([RS_AP_00137](#), [RS_Diag_04150](#), [RS_Diag_04105](#))

8.3.1.10.8 diag::DTCInformation::~~DTCInformation function

[SWS_DM_00665]{DRAFT} [

Kind:	function
Symbol:	~DTCInformation()
Scope:	class ara::diag::DTCInformation
Syntax:	~DTCInformation () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/dtc_information.h"
Description:	Destructor of class DTCInformation.

](RS_AP_00134)

8.3.1.10.9 diag::DTCInformation::GetCurrentStatus function

[SWS_DM_00666]{DRAFT} [

Kind:	function	
Symbol:	GetCurrentStatus(std::uint32_t dtc)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<UdsDtcStatusByteType> GetCurrentStatus (std::uint32_t dtc);	
Parameters (in):	dtc	DTC identifier for which the status should be retrieved.
Return value:	ara::core::Result< UdsDtcStatusByte Type >	the current UDS DTC status byte of the given DTC identifier.
Errors:	DiagErrorDomain::DiagErrc::kNoSuch Dtc	given DTC identifier not available.
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Retrieves the current UDS DTC status byte of the given DTC identifier.	

](RS_AP_00139)

8.3.1.10.10 diag::DTCInformation::SetDTCStatusChangedNotifier function

[SWS_DM_00667]{DRAFT} [

Kind:	function	
Symbol:	SetDTCStatusChangedNotifier(std::function< void(std::uint32_t dtc, UdsDtcStatusByteType udsStatusByteOld, UdsDtcStatusByteType udsStatusByteNew)> notifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> SetDTCStatusChangedNotifier (std::function< void(std::uint32_t dtc, UdsDtcStatusByteType udsStatusByteOld, UdsDtcStatusByteType udsStatusByteNew)> notifier);	





Parameters (in):	notifier	The function to be called if a DTC status has changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kNoSuchDtc	given DTC identifier not available.
	DiagErrorDomain::DiagErrc::kInvalidArgument	given arguments are invalid (pointer, status old/new not plausible).
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Register a notifier function which is called if a UDS DTC status is changed.	

](RS_AP_00139, RS_Diag_04148)

8.3.1.10.11 diag::DTCInformation::SetSnapshotRecordUpdatedNotifier function

[SWS_DM_00668]{DRAFT} [

Kind:	function	
Symbol:	SetSnapshotRecordUpdatedNotifier(std::function< void(SnapshotRecordUpdatedType)> notifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> SetSnapshotRecordUpdatedNotifier (std::function< void(SnapshotRecordUpdatedType)> notifier);	
Parameters (in):	notifier	The function to be called if the SnapshotRecord is changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kInvalidArgument	given argument is invalid (pointer).
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Register a notifier function which is called if the SnapshotRecord is changed.	

](RS_AP_00139, RS_Diag_04205)

8.3.1.10.12 diag::DTCInformation::GetNumberOfStoredEntries function

[SWS_DM_00669]{DRAFT} [

Kind:	function	
Symbol:	GetNumberOfStoredEntries()	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<std::uint32_t> GetNumberOfStoredEntries ();	
Return value:	ara::core::Result< std::uint32_t >	Number of currently stored fault memory entries.
Header file:	#include "ara/diag/dtc_information.h"	





Description:	Contains the number of currently stored fault memory entries.
---------------------	---

](RS_AP_00139, RS_Diag_04109)

8.3.1.10.13 diag::DTCInformation::SetNumberOfStoredEntriesNotifier function

[SWS_DM_00670]{DRAFT} [

Kind:	function	
Symbol:	SetNumberOfStoredEntriesNotifier(std::function< void(std::uint32_t)> notifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> SetNumberOfStoredEntriesNotifier (std::function< void(std::uint32_t)> notifier);	
Parameters (in):	notifier	The function to be called if the number of entries for this diagnostic event memory instance has changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kInvalid Argument	given argument is invalid (pointer).
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Register a notifier function which is called if the number of currently stored fault memory entries changed.	

](RS_AP_00139, RS_Diag_04109)

8.3.1.10.14 diag::DTCInformation::Clear function

[SWS_DM_00671]{DRAFT} [

Kind:	function	
Symbol:	Clear(std::uint32_t DTCGroup)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> Clear (std::uint32_t DTCGroup);	
Parameters (in):	DTCGroup	DTC group to be cleared.
Return value:	ara::core::Result< void >	void or errors
Errors:	DiagErrorDomain::DiagErrc::kBusy	Busy processing.
	DiagErrorDomain::DiagErrc::kFailed	Clear failed.
	DiagErrorDomain::DiagErrc::kMemory Error	Memory error reported.
	DiagErrorDomain::DiagErrc::kWrong Dtc	Wrong DTC group passed.
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Method for Clearing a DTC or a group of DTCs.	

](RS_AP_00119, RS_AP_00139, RS_Diag_04194)

8.3.1.10.15 diag::DTCInformation::GetControlDTCStatus function

[SWS_DM_00672]{DRAFT} [

Kind:	function	
Symbol:	GetControlDTCStatus()	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<ControlDtcStatusType> GetControlDTCStatus ();	
Return value:	ara::core::Result< ControlDtcStatus Type >	The current status of ControlDtcStatus (related to UDS service 0x85)
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Contains the current status of the ControlDTCStatus.	

]([RS_AP_00139](#), [RS_Diag_04159](#))

8.3.1.10.16 diag::DTCInformation::SetControlDtcStatusNotifier function

[SWS_DM_00673]{DRAFT} [

Kind:	function	
Symbol:	SetControlDtcStatusNotifier(std::function< void(ControlDtcStatusType)> notifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> SetControlDtcStatusNotifier (std::function< void(ControlDtcStatusType)> notifier);	
Parameters (in):	notifier	The function to be called if the ControlDTCStatus (related to UDS service 0x85) for this diagnostic memory instance has changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kInvalid Argument	given argument is invalid (pointer).
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Registers a notifier function which is called if the control DTC setting is changed.	

]([RS_AP_00139](#), [RS_Diag_04159](#))

8.3.1.10.17 diag::DTCInformation::EnableControlDtc function

[SWS_DM_00674]{DRAFT} [

Kind:	function	
Symbol:	EnableControlDtc()	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> EnableControlDtc ();	
Return value:	ara::core::Result< void >	–





Header file:	#include "ara/diag/dtc_information.h"
Description:	Enforce restoring ControlDTCStatus setting to enabled in case the monitor has some conditions or states demands to do so.

]([RS_AP_00139](#), [RS_Diag_04159](#))

8.3.1.10.18 diag::DTCInformation::GetEventMemoryOverflow function

[SWS_DM_00919]{DRAFT} [

Kind:	function	
Symbol:	GetEventMemoryOverflow()	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<bool> GetEventMemoryOverflow ();	
Return value:	ara::core::Result< bool >	Current status of event memory overflow.
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Contains the current event memory overflow status.	

]([RS_AP_00139](#), [RS_Diag_04093](#))

8.3.1.10.19 diag::DTCInformation::SetEventMemoryOverflowNotifier function

[SWS_DM_00918]{DRAFT} [

Kind:	function	
Symbol:	SetEventMemoryOverflowNotifier(std::function< void(bool)> notifier)	
Scope:	class ara::diag::DTCInformation	
Syntax:	ara::core::Result<void> SetEventMemoryOverflowNotifier (std::function< void(bool)> notifier);	
Parameters (in):	notifier	The function to be called if the overflow status for this diagnostic event memory instance has changed.
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagErrc::kInvalid Argument	given argument is invalid (pointer).
Header file:	#include "ara/diag/dtc_information.h"	
Description:	Register a notifier function which is called if the current event memory overflow status changed.	

]([RS_AP_00139](#), [RS_Diag_04093](#))

8.3.1.11 Conversation class

The conversation object can only be retrieved by a given meta_info object.

[SWS_DM_00693]{DRAFT} [

Kind:	class
Symbol:	Conversation
Scope:	namespace ara::diag
Syntax:	class Conversation {...};
Header file:	#include "ara/diag/conversation.h"
Description:	Conversation interface.

]([RS_Diag_04166](#), [RS_Diag_04169](#))

8.3.1.11.1 diag::ActivityStatusType type

[SWS_DM_00690]{DRAFT} [

Kind:	enumeration	
Symbol:	ActivityStatusType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class ActivityStatusType : std::uint8_t {...};	
Values:	kActive= 0x00	Currently active; i.e. request is currently processed or non-default session is active.
	kInactive= 0x01	Currently not active.
Header file:	#include "ara/diag/conversation.h"	
Description:	Type for current activity status.	

]([RS_AP_00125](#), [RS_Diag_04166](#), [RS_Diag_04169](#), [RS_Diag_04209](#))

8.3.1.11.2 diag::SessionControlType type

[SWS_DM_00706]{DRAFT} [

Kind:	enumeration	
Symbol:	SessionControlType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class SessionControlType : std::uint8_t {...};	
Values:	kDefaultSession= 0x01	default session according to ISO 14229-1.
	kProgrammingSession= 0x02	programming session according to ISO 14229-1.
	kExtendedDiagnosticSession= 0x03	extended diagnostic session according to ISO 14229-1.
	kSafetySystemDiagnosticSession= 0x04	configuration dependent sessionssafety system diagnostic session according to ISO 14229-1.



△

Header file:	#include "ara/diag/conversation.h"
Description:	Type for the active diagnostic session. .

] ([RS_AP_00125](#))

8.3.1.11.3 diag::SecurityLevelType type

[SWS_DM_00705]{DRAFT} [

Kind:	enumeration	
Symbol:	SecurityLevelType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class SecurityLevelType : std::uint8_t {...};	
Values:	kLocked= 0x00	security level locked;
	configuration_dependent	i.e. the security is enabled, configuration dependent; formula: (SecurityAccessType (requestSeed) + 1) / 2
Header file:	#include "ara/diag/conversation.h"	
Description:	Type for the active security level. .	

] ([RS_AP_00125](#))

8.3.1.11.4 diag::Conversation::ConversationIdentifierType type

[SWS_DM_00691]{DRAFT} [

Kind:	struct
Symbol:	ConversationIdentifierType
Scope:	class ara::diag::Conversation
Syntax:	struct ConversationIdentifierType {...};
Header file:	#include "ara/diag/conversation.h"
Description:	Properties allowing an identification of the conversation.

] ([RS_Diag_04166](#), [RS_Diag_04169](#), [RS_Diag_04209](#))

8.3.1.11.5 diag::Conversation::GetConversation function

[SWS_DM_00692]{DRAFT} [

Kind:	function	
Symbol:	GetConversation(MetaInfo &metaInfo)	
Scope:	class ara::diag::Conversation	
Syntax:	static ara::core::Result<Conversation&> GetConversation (MetaInfo &metaInfo);	
Parameters (in):	metaInfo	contains additional meta information
Return value:	ara::core::Result< Conversation & >	Conversation object or error
Header file:	#include "ara/diag/conversation.h"	
Description:	Get one conversation based on given MetaInfo.	

|(RS_AP_00139, RS_Diag_04169, RS_Diag_04170)

8.3.1.11.6 diag::Conversation::GetAllConversations function

[SWS_DM_00782]{DRAFT} [

Kind:	function	
Symbol:	GetAllConversations()	
Scope:	class ara::diag::Conversation	
Syntax:	static ara::core::Vector<std::reference_wrapper<Conversation> > GetAllConversations ();	
Return value:	ara::core::Vector< std::reference_wrapper< Conversation > >	a vector of all possible Conversation objects
Header file:	#include "ara/diag/conversation.h"	
Description:	Get all possible conversations.	

|(RS_Diag_04166, RS_Diag_04169, RS_Diag_04209)

8.3.1.11.7 diag::Conversation::GetCurrentActiveConversations function

[SWS_DM_00783]{DRAFT} [

Kind:	function	
Symbol:	GetCurrentActiveConversations()	
Scope:	class ara::diag::Conversation	
Syntax:	static ara::core::Vector<std::reference_wrapper<Conversation> > GetCurrentActiveConversations ();	
Return value:	ara::core::Vector< std::reference_wrapper< Conversation > >	a vector of all currently active (GetActivityStatus() == kActive) Conversation objects
Header file:	#include "ara/diag/conversation.h"	
Description:	Get all currently active conversations.	

|(RS_Diag_04166, RS_Diag_04169, RS_Diag_04209)

8.3.1.11.8 diag::Conversation::GetActivityStatus function

[SWS_DM_00694]{DRAFT} [

Kind:	function	
Symbol:	GetActivityStatus()	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<ActivityStatusType> GetActivityStatus ();	
Return value:	ara::core::Result< ActivityStatusType >	the activity status of the conversation
Header file:	#include "ara/diag/conversation.h"	
Description:	Represents the status of an active conversation.	

]([RS_AP_00139](#), [RS_Diag_04169](#))

8.3.1.11.9 diag::Conversation::SetActivityNotifier function

[SWS_DM_00695]{DRAFT} [

Kind:	function	
Symbol:	SetActivityNotifier(std::function< void(ActivityStatusType)> notifier)	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<void> SetActivityNotifier (std::function< void(ActivityStatusType)> notifier);	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	void when the registering went fine or error
Header file:	#include "ara/diag/conversation.h"	
Description:	Register a notifier function which is called if the activity is changed.	

]([RS_AP_00139](#), [RS_Diag_04169](#))

8.3.1.11.10 diag::Conversation::GetConversationIdentifier function

[SWS_DM_00700]{DRAFT} [

Kind:	function	
Symbol:	GetConversationIdentifier()	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<ConversationIdentifierType> GetConversation Identifier ();	
Return value:	ara::core::Result< Conversation IdentifierType >	the conversation information
Header file:	#include "ara/diag/conversation.h"	
Description:	Getter for the current identification properties of the active conversation.	

]([RS_AP_00139](#), [RS_Diag_04169](#))

8.3.1.11.11 diag::Conversation::GetDiagnosticSession function

[SWS_DM_00696]{DRAFT} [

Kind:	function	
Symbol:	GetDiagnosticSession()	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<SessionControlType> GetDiagnosticSession ();	
Return value:	ara::core::Result< SessionControlType >	the current session
Header file:	#include "ara/diag/conversation.h"	
Description:	Represents the current active diagnostic session of an active conversation.	

]([RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.12 diag::Conversation::SetDiagnosticSessionNotifier function

[SWS_DM_00697]{DRAFT} [

Kind:	function	
Symbol:	SetDiagnosticSessionNotifier(std::function< void(SessionControlType)> notifier)	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<void> SetDiagnosticSessionNotifier (std::function< void(SessionControlType)> notifier);	
Parameters (in):	notifier	notifier function to be called
Return value:	ara::core::Result< void >	void when the registering went fine or error
Header file:	#include "ara/diag/conversation.h"	
Description:	Register a notifier function which is called if the Session is changed.	

]([RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.13 diag::Conversation::GetDiagnosticSessionShortName function

[SWS_DM_00707]{DRAFT} [

Kind:	function	
Symbol:	GetDiagnosticSessionShortName(SessionControlType session)	
Scope:	class ara::diag::Conversation	
Syntax:	ara::core::Result<ara::core::StringView> GetDiagnosticSessionShortName (SessionControlType session);	
Parameters (in):	session	Diagnostic session the shortname shall be returned for.
Return value:	ara::core::Result< ara::core::String View >	ara::core::Result<ara::core::StringView> the session as shortName; DiagnosticSession.shortName





Header file:	#include "ara/diag/conversation.h"
Description:	Converts the given diagnostic session into the ShortName.

]([RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.14 diag::Conversation::GetDiagnosticSecurityLevel function

[SWS_DM_00698]{DRAFT} [

Kind:	function
Symbol:	GetDiagnosticSecurityLevel()
Scope:	class ara::diag::Conversation
Syntax:	ara::core::Result<SecurityLevelType> GetDiagnosticSecurityLevel ();
Return value:	ara::core::Result< SecurityLevelType > the current SecurityLevel
Header file:	#include "ara/diag/conversation.h"
Description:	Represents the current active diagnostic SecurityLevel of an active conversation.

]([RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.15 diag::Conversation::SetSecurityLevelNotifier function

[SWS_DM_00699]{DRAFT} [

Kind:	function
Symbol:	SetSecurityLevelNotifier(std::function< void(SecurityLevelType)> notifier)
Scope:	class ara::diag::Conversation
Syntax:	ara::core::Result<void> SetSecurityLevelNotifier (std::function< void(SecurityLevelType)> notifier);
Parameters (in):	notifier notifier function to be called
Return value:	ara::core::Result< void > void when the registering went fine or error
Header file:	#include "ara/diag/conversation.h"
Description:	Register a notifier function which is called if the SecurityLevel is changed.

]([RS_AP_00139](#), [RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.16 diag::Conversation::GetDiagnosticSecurityLevelShortName function

[SWS_DM_00708]{DRAFT} [

Kind:	function	
Symbol:	GetDiagnosticSecurityLevelShortName(SecurityLevelType securityLevel)	
Scope:	class ara::diag::Conversation	
Syntax:	<pre>ara::core::Result<ara::core::StringView> GetDiagnosticSecurityLevelShortName (SecurityLevelType securityLevel);</pre>	
Parameters (in):	securityLevel	Security level enum the shortname shall be returned for.
Return value:	ara::core::Result< ara::core::StringView >	ara::core::Result<ara::core::StringView> the SecurityLevel as shortName; DiagnosticSecurityLevel.shortName
Header file:	#include "ara/diag/conversation.h"	
Description:	Converts the given diagnostic SecurityLevel into the ShortName.	

]([RS_Diag_04169](#), [RS_Diag_04208](#))

8.3.1.11.17 diag::Conversation::ResetToDefaultSession function

[SWS_DM_00701]{DRAFT} [

Kind:	function	
Symbol:	ResetToDefaultSession()	
Scope:	class ara::diag::Conversation	
Syntax:	<pre>ara::core::Result<void> ResetToDefaultSession ();</pre>	
Return value:	ara::core::Result< void >	void on success or error
Header file:	#include "ara/diag/conversation.h"	
Description:	Method to reset the current session to default session.	

]([RS_Diag_04006](#), [RS_Diag_04166](#), [RS_Diag_04169](#), [RS_Diag_04209](#))

8.3.1.12 Condition class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticConditionInterface](#).

[SWS_DM_00711]{DRAFT} [

Kind:	class	
Symbol:	Condition	
Scope:	namespace ara::diag	
Syntax:	<pre>class Condition {...};</pre>	
Header file:	#include "ara/diag/condition.h"	
Description:	DiagnosticConditionInterface.	

]([RS_Diag_04192](#))

8.3.1.12.1 diag::ConditionType type

[SWS_DM_00710]{DRAFT} [

Kind:	enumeration	
Symbol:	ConditionType	
Scope:	namespace ara::diag	
Underlying type:	–	
Syntax:	enum class ConditionType {...};	
Values:	kConditionFalse= 0x00	condition is set to false
	kConditionTrue= 0x01	condition is set to true
Header file:	#include "ara/diag/condition.h"	
Description:	Type for Condition status.	

]([RS_AP_00125](#), [RS_Diag_04192](#))

8.3.1.12.2 diag::Condition::Condition function

[SWS_DM_00712]{DRAFT} [

Kind:	function	
Symbol:	Condition(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::Condition	
Syntax:	explicit Condition (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticConditionInterface
Header file:	#include "ara/diag/condition.h"	
Description:	Constructor of Condition Class.	

]([RS_AP_00137](#), [RS_Diag_04192](#))

8.3.1.12.3 diag::Condition::~~Condition function

[SWS_DM_00713]{DRAFT} [

Kind:	function	
Symbol:	~Condition()	
Scope:	class ara::diag::Condition	
Syntax:	~Condition () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/condition.h"	
Description:	Destructor of class Condition.	

]([RS_AP_00134](#), [RS_Diag_04192](#))

8.3.1.12.4 diag::Condition::GetCondition function

[SWS_DM_00714]{DRAFT} [

Kind:	function	
Symbol:	GetCondition()	
Scope:	class ara::diag::Condition	
Syntax:	ara::core::Result<ConditionType> GetCondition ();	
Return value:	ara::core::Result< ConditionType >	the current condition
Header file:	#include "ara/diag/condition.h"	
Description:	Get current condition.	

]([RS_AP_00139](#), [RS_Diag_04192](#))

8.3.1.12.5 diag::Condition::SetCondition function

[SWS_DM_00715]{DRAFT} [

Kind:	function	
Symbol:	SetCondition(ConditionType condition)	
Scope:	class ara::diag::Condition	
Syntax:	ara::core::Result<void> SetCondition (ConditionType condition);	
Parameters (in):	condition	current condition
Return value:	ara::core::Result< void >	–
Header file:	#include "ara/diag/condition.h"	
Description:	Set condition.	

]([RS_AP_00139](#), [RS_Diag_04192](#))

8.3.1.13 OperationCycle class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticOperationCycleInterface](#).

[SWS_DM_00751]{DRAFT} [

Kind:	class	
Symbol:	OperationCycle	
Scope:	namespace ara::diag	
Syntax:	class OperationCycle {...};	
Header file:	#include "ara/diag/operation_cycle.h"	
Description:	DiagnosticOperationCycleInterface provides functionality for handling of operation cycles.	

]([RS_Diag_04178](#))

8.3.1.13.1 diag::OperationCycle::OperationCycle function

[SWS_DM_00752]{DRAFT} [

Kind:	function	
Symbol:	OperationCycle(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::OperationCycle	
Syntax:	explicit OperationCycle (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticOperationCycleInterface
Header file:	#include "ara/diag/operation_cycle.h"	
Description:	Constructor for DiagnosticOperationCycleInterface.	

]([RS_AP_00137](#), [RS_Diag_04178](#))

8.3.1.13.2 diag::OperationCycle::~~OperationCycle function

[SWS_DM_00753]{DRAFT} [

Kind:	function	
Symbol:	~OperationCycle()	
Scope:	class ara::diag::OperationCycle	
Syntax:	~OperationCycle () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/operation_cycle.h"	
Description:	Destructor of DiagnosticOperationCycleInterface.	

]([RS_AP_00134](#), [RS_Diag_04178](#))

8.3.1.13.3 diag::OperationCycle::SetNotifier function

[SWS_DM_00755]{DRAFT} [

Kind:	function	
Symbol:	SetNotifier(std::function< void()> notifier)	
Scope:	class ara::diag::OperationCycle	
Syntax:	ara::core::Result<void> SetNotifier (std::function< void()> notifier);	
DIRECTION NOT DEFINED	notifier	–
Return value:	ara::core::Result< void >	–
Header file:	#include "ara/diag/operation_cycle.h"	
Description:	Registering a notifier function which is called if the operation cycle is changed.	

]([RS_AP_00139](#), [RS_Diag_04178](#), [RS_Diag_04186](#))

8.3.1.13.4 diag::OperationCycle::RestartOperationCycle function

[SWS_DM_01102]{DRAFT} [

Kind:	function	
Symbol:	RestartOperationCycle()	
Scope:	class ara::diag::OperationCycle	
Syntax:	ara::core::Result<void> RestartOperationCycle ();	
Return value:	ara::core::Result< void >	–
Header file:	#include "ara/diag/operation_cycle.h"	
Description:	Trigger to restart the OperationCycle.	

]([RS_AP_00139](#), [RS_Diag_04178](#), [RS_Diag_04182](#))

8.3.1.14 Indicator class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticIndicatorInterface](#).

[SWS_DM_00741]{DRAFT} [

Kind:	class	
Symbol:	Indicator	
Scope:	namespace ara::diag	
Syntax:	class Indicator {...};	
Header file:	#include "ara/diag/indicator.h"	
Description:	DiagnosticIndicatorInterface provides functionality for handling indicators.	

]([RS_Diag_04204](#))

8.3.1.14.1 diag::IndicatorType type

[SWS_DM_00740]{DRAFT} [

Kind:	enumeration	
Symbol:	IndicatorType	
Scope:	namespace ara::diag	
Underlying type:	–	
Syntax:	enum class IndicatorType {...};	
Values:	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.
Header file:	#include "ara/diag/indicator.h"	
Description:	Represents the state of an indicator.	

|(RS_Diag_04204)

8.3.1.14.2 diag::Indicator::Indicator function

[SWS_DM_00742]{DRAFT} [

Kind:	function	
Symbol:	Indicator(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::Indicator	
Syntax:	explicit Indicator (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticIndicatorInterface
Header file:	#include "ara/diag/indicator.h"	
Description:	Constructor for DiagnosticIndicatorInterface.	

|(RS_AP_00137, RS_Diag_04204)

8.3.1.14.3 diag::Indicator::~~Indicator function

[SWS_DM_00743]{DRAFT} [

Kind:	function	
Symbol:	~Indicator()	
Scope:	class ara::diag::Indicator	
Syntax:	~Indicator () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/indicator.h"	
Description:	Destructor of DiagnosticIndicatorInterface.	

|(RS_AP_00134, RS_Diag_04204)

8.3.1.14.4 diag::Indicator::GetIndicator function

[SWS_DM_00744]{DRAFT} [

Kind:	function	
Symbol:	GetIndicator()	
Scope:	class ara::diag::Indicator	
Syntax:	ara::core::Result<IndicatorType> GetIndicator ();	
Return value:	ara::core::Result< IndicatorType >	the current Indicator
Header file:	#include "ara/diag/indicator.h"	
Description:	Get current Indicator.	

|(RS_AP_00139, RS_Diag_04204)

8.3.1.14.5 diag::Indicator::SetNotifier function

[SWS_DM_00745]{DRAFT} [

Kind:	function	
Symbol:	SetNotifier(std::function< void(IndicatorType)> notifier)	
Scope:	class ara::diag::Indicator	
Syntax:	ara::core::Result<void> SetNotifier (std::function< void(IndicatorType)> notifier);	
Parameters (in):	notifier	notifier function
Return value:	ara::core::Result< void >	–
Header file:	#include "ara/diag/indicator.h"	
Description:	Register a notifier function which is called if the indicator is updated.	

|(RS_AP_00139, RS_Diag_04204)

8.3.1.15 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[2] as a Manufacturer or a Supplier Check.

[SWS_DM_00771]{DRAFT} [

Kind:	class	
Symbol:	ServiceValidation	
Scope:	namespace ara::diag	
Syntax:	class ServiceValidation {...};	





Header file:	#include "ara/diag/service_validation.h"
Description:	DiagnosticServiceValidationInterface.

]([RS_Diag_04199](#))

8.3.1.15.1 diag::ConfirmationStatusType

[SWS_DM_00770]{DRAFT} [

Kind:	enumeration	
Symbol:	ConfirmationStatusType	
Scope:	namespace ara::diag	
Underlying type:	std::uint8_t	
Syntax:	enum class ConfirmationStatusType : std::uint8_t {...};	
Values:	kResPosOk= 0x00	Positive response has been sent out successfully.
	kResPosNotOk= 0x01	Positive response has not been sent out successfully.
	kResNegOk= 0x02	Negative response has been sent out successfull.
	kResNegNotOk= 0x03	Negative response has not been sent out successfully.
	kResPosSuppressed= 0x04	Positive answer suppressed.
	kResNegSuppressed= 0x05	Negative answer suppressed.
	kCanceled= 0x06	Processing is canceled.
	kNoProcessingNoResponse= 0x07	Processing rejected in Validation.
Header file:	#include "ara/diag/service_validation.h"	
Description:	Represents the status of the service processing.	

]([RS_Diag_04199](#))

8.3.1.15.2 diag::ServiceValidation::ServiceValidation function

[SWS_DM_00772]{DRAFT} [

Kind:	function	
Symbol:	ServiceValidation(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::ServiceValidation	
Syntax:	explicit ServiceValidation (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticServiceValidationInterface
Header file:	#include "ara/diag/service_validation.h"	
Description:	Constructor of ServiceValidation.	

]([RS_AP_00137](#), [RS_Diag_04199](#))

8.3.1.15.3 diag::ServiceValidation::~~ServiceValidation function

[SWS_DM_00773]{DRAFT} [

Kind:	function
Symbol:	~ServiceValidation()
Scope:	class ara::diag::ServiceValidation
Syntax:	virtual ~ServiceValidation () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/service_validation.h"
Description:	Destructor of ServiceValidation.

]([RS_AP_00134](#), [RS_Diag_04199](#))

8.3.1.15.4 diag::ServiceValidation::Validate function

[SWS_DM_00774]{DRAFT} [

Kind:	function	
Symbol:	Validate(ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo)	
Scope:	class ara::diag::ServiceValidation	
Syntax:	virtual ara::core::Future<void> Validate (ara::core::Span< std::uint8_t > requestData, MetaInfo &metaInfo);	
Parameters (in):	requestData	Diagnostic request data (including SID).
	metaInfo	MetaInfo of the request.
Return value:	ara::core::Future< void >	Returns nothing or an error
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.



△

	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoProcessingNoResponse	0xFF, ///< Deviating from ISO - no further service processing and no response (silently ignore request message).
Header file:	#include "ara/diag/service_validation.h"	
Description:	Called for any request message.	

]([RS_AP_00138](#), [RS_Diag_04170](#), [RS_Diag_04199](#), [RS_AP_00119](#))

8.3.1.15.5 diag::ServiceValidation::Confirmation function

[SWS_DM_00775]{DRAFT} [

Kind:	function	
Symbol:	Confirmation(ConfirmationStatusType status, MetaInfo &metaInfo)	
Scope:	class ara::diag::ServiceValidation	
Syntax:	virtual ara::core::Future<void> Confirmation (ConfirmationStatusType status, MetaInfo &metaInfo);	
Parameters (in):	status	status/outcome of the service processing.
	metaInfo	MetaInfo of the request.
Return value:	ara::core::Future< void >	Returns nothing or an error
Header file:	#include "ara/diag/service_validation.h"	
Description:	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](#))

8.3.1.15.6 diag::ServiceValidation::Offer function

[SWS_DM_00776]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::ServiceValidation	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	Returns nothing or an error
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/service_validation.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00119, RS_AP_00139, RS_Diag_04199)

8.3.1.15.7 diag::ServiceValidation::StopOffer function

[SWS_DM_00777]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class ara::diag::ServiceValidation	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/service_validation.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

|(RS_Diag_04199)

8.3.1.16 SecurityAccess class

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticSecurityLevelInterface](#).

[SWS_DM_00761]{DRAFT} [

Kind:	class	
Symbol:	SecurityAccess	
Scope:	namespace ara::diag	
Syntax:	class SecurityAccess {...};	
Header file:	#include "ara/diag/security_access.h"	
Description:	DiagnosticSecurityAccessInterface.	

|(RS_Diag_04005)

8.3.1.16.1 diag::KeyCompareResultType type

[SWS_DM_00760]{DRAFT} [

Kind:	enumeration	
Symbol:	KeyCompareResultType	
Scope:	namespace ara::diag	
Underlying type:	–	
Syntax:	enum class KeyCompareResultType {...};	
Values:	kKeyValid= 0x00	Key is valid.
	kKeyInvalid= 0x01	Key is invalid.
Header file:	#include "ara/diag/security_access.h"	
Description:	Represents the status of the key compare.	

]([RS_Diag_04005](#))

8.3.1.16.2 diag::SecurityAccess::SecurityAccess function

[SWS_DM_00762]{DRAFT} [

Kind:	function	
Symbol:	SecurityAccess(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::SecurityAccess	
Syntax:	explicit SecurityAccess (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticSecurityAccessInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/security_access.h"	
Description:	Constructor of SecurityAccess.	

]([RS_AP_00137](#), [RS_Diag_04005](#))

8.3.1.16.3 diag::SecurityAccess::~~SecurityAccess function

[SWS_DM_00763]{DRAFT} [

Kind:	function	
Symbol:	~SecurityAccess()	
Scope:	class ara::diag::SecurityAccess	
Syntax:	virtual ~SecurityAccess () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/security_access.h"	





Description:	Destructor of SecurityAccess.
---------------------	-------------------------------

](RS_AP_00134, RS_Diag_04005)

8.3.1.16.4 diag::SecurityAccess::GetSeed function

[SWS_DM_00764]{DRAFT} [

Kind:	function	
Symbol:	GetSeed(ara::core::Span< std::uint8_t > securityAccessDataRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::SecurityAccess	
Syntax:	virtual ara::core::Future<ara::core::Span<std::uint8_t> > GetSeed (ara::core::Span< std::uint8_t > securityAccessDataRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	securityAccessDataRecord	Security Access payload
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< ara::core::Span< std::uint8_t > >	provided seed
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/security_access.h"	
Description:	Called for any request message.	

](RS_AP_00138, RS_Diag_04005, RS_Diag_04170, RS_AP_00119)

8.3.1.16.5 diag::SecurityAccess::CompareKey function

[SWS_DM_00765]{DRAFT} [

Kind:	function	
Symbol:	CompareKey(ara::core::Span< std::uint8_t > key, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::SecurityAccess	
Syntax:	virtual ara::core::Future<KeyCompareResultType> CompareKey (ara::core::Span< std::uint8_t > key, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	key	The key to be validated
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< KeyCompareResult Type >	Result of the key validation.
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kIncorrectMessageLengthOr InvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/security_access.h"	
Description:	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](#))

8.3.1.16.6 diag::SecurityAccess::Offer function

[SWS_DM_00766]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::SecurityAccess	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/security_access.h"	





Description:	This Offer will enable the DM to forward request messages to this handler.
---------------------	--

]([RS_AP_00139](#), [RS_Diag_04005](#), [RS_AP_00119](#))

8.3.1.16.7 diag::SecurityAccess::StopOffer function

[SWS_DM_00767]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::SecurityAccess
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/security_access.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

]([RS_Diag_04005](#))

8.3.1.17 CommunicationControl class

[SWS_DM_00804]{DRAFT} [

Kind:	class
Symbol:	CommunicationControl
Scope:	namespace ara::diag
Syntax:	class CommunicationControl {...};
Header file:	#include "ara/diag/communication_control.h"
Description:	CommunicationControl interface.

]([RS_Diag_04196](#))

8.3.1.17.1 diag::CommunicationControl::ComCtrlRequestParamsType type

[SWS_DM_00805]{DRAFT} [

Kind:	struct
Symbol:	ComCtrlRequestParamsType
Scope:	class ara::diag::CommunicationControl
Syntax:	struct ComCtrlRequestParamsType {...};





Header file:	#include "ara/diag/communication_control.h"
Description:	ComCtrlRequestParamsType is a structure, which holds all parameters of an UDS 0x28 communicationControl request.

|(RS_Diag_04196) CommunicationControl

8.3.1.17.2 diag::CommunicationControl::CommunicationControl function

[SWS_DM_00806]{DRAFT} [

Kind:	function	
Symbol:	CommunicationControl(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::CommunicationControl	
Syntax:	explicit CommunicationControl (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticCommunicationControlInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/communication_control.h"	
Description:	Class for an CommunicationControl.	

|(RS_AP_00137, RS_Diag_04196)

8.3.1.17.3 diag::CommunicationControl::~~CommunicationControl function

[SWS_DM_00807]{DRAFT} [

Kind:	function	
Symbol:	~CommunicationControl()	
Scope:	class ara::diag::CommunicationControl	
Syntax:	virtual ~CommunicationControl () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/communication_control.h"	
Description:	Destructor of class CommunicationControl.	

|(RS_AP_00134, RS_Diag_04196)

8.3.1.17.4 diag::CommunicationControl::CommCtrlRequest function

[SWS_DM_00808]{DRAFT} [

Kind:	function	
Symbol:	CommCtrlRequest(ComCtrlRequestParamsType controlType, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::CommunicationControl	
Syntax:	virtual ara::core::Future<void> CommCtrlRequest (ComCtrlRequestParamsType controlType, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	controlType	All UDS request parameters packed into a structure since it holds optional elements
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< void >	–
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/communication_control.h"	
Description:	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)

8.3.1.17.5 diag::CommunicationControl::Offer function

[SWS_DM_00809]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::CommunicationControl	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReportingErrc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/communication_control.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00119, RS_AP_00139, RS_Diag_04196)

8.3.1.17.6 diag::CommunicationControl::StopOffer function

[SWS_DM_00810]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::CommunicationControl
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/communication_control.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

]([RS_Diag_04196](#))

8.3.1.18 DownloadService class

This interface is newly introduced.

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [Diagnostic-DownloadInterface](#).

[SWS_DM_00784]{DRAFT} [

Kind:	class
Symbol:	DownloadService
Scope:	namespace ara::diag
Syntax:	class DownloadService {...};
Header file:	#include "ara/diag/download.h"
Description:	Download service interface.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.18.1 diag::DownloadService::OperationOutput type

[SWS_DM_00785]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class ara::diag::DownloadService
Syntax:	struct OperationOutput {...};
Header file:	#include "ara/diag/download.h"
Description:	Response data of positive response message.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

[SWS_DM_00786]{DRAFT} [

Kind:	variable
Symbol:	responseData
Scope:	struct ara::diag::DownloadService::OperationOutput
Type:	ara::core::Vector< std::uint8_t >
Syntax:	ara::core::Vector<std::uint8_t> responseData;
Header file:	#include "ara/diag/download.h"
Description:	Content of positive response message (without SID) Depending on the operation (e.g.: DownloadData, RequestDownloadExit) the expectation, what responseData shall contain (where it starts in the positive response) might differ. See doc of corresponding operation.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.18.2 diag::DownloadService::DownloadService function

[SWS_DM_00787]{DRAFT} [

Kind:	function	
Symbol:	DownloadService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::DownloadService	
Syntax:	explicit DownloadService (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DownloadServiceInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/download.h"	
Description:	Class for an DownloadService.	

]([RS_AP_00137](#), [RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.18.3 diag::DownloadService::~~DownloadService function

[SWS_DM_00788]{DRAFT} [

Kind:	function
Symbol:	~DownloadService()
Scope:	class ara::diag::DownloadService
Syntax:	virtual ~DownloadService () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/download.h"
Description:	Destructor of class DownloadService.

]([RS_AP_00134](#), [RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.18.4 diag::DownloadService::RequestDownload function

[SWS_DM_00789]{DRAFT} [

Kind:	function	
Symbol:	RequestDownload(std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< std::uint8_t > memoryAddressAndSize, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::DownloadService	
Syntax:	virtual ara::core::Future<void> RequestDownload (std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< std::uint8_t > memoryAddressAndSize, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	dataFormatIdentifier	UDS dataFormat Identifier
	addressAndLengthFormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< void >	a Future, which either gets readied to void (for a positive response message) or readied with Error Code from DiagUdsNrcErrc (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/download.h"	
Description:	Called for RequestDownload.	

](RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196)

8.3.1.18.5 diag::DownloadService::DownloadData function

[SWS_DM_00790]{DRAFT} [

Kind:	function
Symbol:	DownloadData(ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)
Scope:	class ara::diag::DownloadService





Syntax:	virtual ara::core::Future<OperationOutput> DownloadData (ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	transferRequestParameterRecord	data to be transferred (copied/downloaded to the ECU/server).
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to Operation Output (transferResponseParameterRecord for a positive response message) or readied with Error Code from DiagUdsNrcErrc (for an negative response message). Data in Operation Output.responseData will be placed after block SequenceCounter as transferResponseParameter Record in the positive response.
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/download.h"	
Description:	Called for TransferData following a previous RequestDownload.	

|(RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196)

8.3.1.18.6 diag::DownloadService::RequestDownloadExit function

[SWS_DM_00791]{DRAFT} [

Kind:	function
Symbol:	RequestDownloadExit(ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)
Scope:	class ara::diag::DownloadService
Syntax:	virtual ara::core::Future<OperationOutput> RequestDownloadExit (ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;





Parameters (in):	transferRequestParameterRecord	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific.
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to Operation Output (transferResponseParameterRecord for a positive response message) or readied with Error Code from DiagUdsNrcErrc (for a negative response message) Data in Operation Output.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
Errors:	Any	applicable NegativeResponseValue according to DiagUdsNrcErrc
Header file:	#include "ara/diag/download.h"	
Description:	Called for RequestTransferExit.	

](RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196)

8.3.1.18.7 diag::DownloadService::Offer function

[SWS_DM_00792]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::DownloadService	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReportingErrc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/download.h"	
Description:	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)

8.3.1.18.8 diag::DownloadService::StopOffer function

[SWS_DM_00793]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::DownloadService
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/download.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.19 UploadService class

This interface is newly introduced.

The *InstanceSpecifier* is only compatible with [PortInterface](#) of [DiagnosticUploadInterface](#).

[SWS_DM_00794]{DRAFT} [

Kind:	class
Symbol:	UploadService
Scope:	namespace ara::diag
Syntax:	class UploadService {...};
Header file:	#include "ara/diag/upload.h"
Description:	Upload service interface.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.19.1 diag::UploadService::OperationOutput type

[SWS_DM_00795]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class ara::diag::UploadService
Syntax:	struct OperationOutput {...};
Header file:	#include "ara/diag/upload.h"
Description:	Response data of positive response message.

]([RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.19.2 diag::UploadService::UploadService function

[SWS_DM_00797]{DRAFT} [

Kind:	function	
Symbol:	UploadService(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::UploadService	
Syntax:	explicit UploadService (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DownloadServiceInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/upload.h"	
Description:	Class for an UploadService.	

]([RS_AP_00137](#), [RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.19.3 diag::UploadService::~~UploadService function

[SWS_DM_00798]{DRAFT} [

Kind:	function	
Symbol:	~UploadService()	
Scope:	class ara::diag::UploadService	
Syntax:	virtual ~UploadService () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/upload.h"	
Description:	Destructor of class UploadService.	

]([RS_AP_00134](#), [RS_Diag_04033](#), [RS_Diag_04196](#))

8.3.1.19.4 diag::UploadService::RequestUpload function

[SWS_DM_00799]{DRAFT} [

Kind:	function	
Symbol:	RequestUpload(std::uint8_t dataFormatIdentifier, std::uint8_t addressAndLengthFormat Identifier, ara::core::Span< std::uint8_t > memoryAddressAndSize, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::UploadService	
Syntax:	virtual ara::core::Future<void> RequestUpload (std::uint8_t dataFormat Identifier, std::uint8_t addressAndLengthFormatIdentifier, ara::core::Span< std::uint8_t > memoryAddressAndSize, MetaInfo &meta Info, CancellationHandler cancellationHandler)=0;	
Parameters (in):	dataFormatIdentifier	UDS dataFormat Identifier





	addressAndLengthFormatIdentifier	UDS addressAndLengthFormatIdentifier
	memoryAddressAndSize	memoryAddress and memorySize part of the request
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/upload.h"	
Description:	Called for RequestDownload.	

|(RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196)

8.3.1.19.5 diag::UploadService::UploadData function

[SWS_DM_00800]{DRAFT} [

Kind:	function	
Symbol:	UploadData(std::size_t numBytesToReturn, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::UploadService	
Syntax:	virtual ara::core::Future<OperationOutput> UploadData (std::size_t numBytesToReturn, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	numBytesToReturn	number of bytes DM accepts (due to its internal buffer) for this chunk.
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to Operation Output (transferResponseParameterRecord for a positive response message) or readied with Error Code from DiagUdsNrcErrc (for an negative response message). Data in Operation Output.responseData will be placed after block SequenceCounter as transferResponseParameter Record in the positive response.





Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/upload.h"	
Description:	Called for TransferData following a previous RequestUpload.	

]([RS_AP_00119](#), [RS_AP_00138](#), [RS_Diag_04033](#), [RS_Diag_04170](#), [RS_Diag_04196](#))

8.3.1.19.6 diag::UploadService::RequestUploadExit function

[SWS_DM_00801]{DRAFT} [

Kind:	function	
Symbol:	RequestUploadExit(ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::UploadService	
Syntax:	virtual ara::core::Future<OperationOutput> RequestUploadExit (ara::core::Span< std::uint8_t > transferRequestParameterRecord, MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	transferRequestParameterRecord	This parameter record contains parameter(s), which are required by the server to support the transfer of data. Format and length of this parameter(s) are vehicle manufacturer specific.
	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Future, which either gets readied to Operation Output (transferResponseParameterRecord for a positive response message) or readied with Error Code from DiagUdsNrcErrc (for an negative response message) Data in Operation Output.responseData will be placed after SID as transferResponseParameterRecord in the positive response.
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/upload.h"	
Description:	Called for RequestTransferExit.	

|(RS_AP_00119, RS_AP_00138, RS_Diag_04033, RS_Diag_04170, RS_Diag_04196)

8.3.1.19.7 diag::UploadService::Offer function

[SWS_DM_00802]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::UploadService	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReportingErrc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::kAlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/upload.h"	
Description:	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)

8.3.1.19.8 diag::UploadService::StopOffer function

[SWS_DM_00803]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class ara::diag::UploadService	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/upload.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

|(RS_Diag_04033, RS_Diag_04196)

8.3.1.20 EcuResetRequest class

[SWS_DM_01009]{DRAFT} [

Kind:	class
Symbol:	EcuResetRequest
Scope:	namespace ara::diag
Syntax:	<code>class EcuResetRequest {...};</code>
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>
Description:	Service EcuReset Request interface.

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.20.1 diag::EcuResetRequest::ResetRequestType type

[SWS_DM_01007]{DRAFT} [

Kind:	enumeration								
Symbol:	ResetRequestType								
Scope:	namespace ara::diag								
Underlying type:	std::uint32_t								
Syntax:	<code>enum class ResetRequestType : std::uint32_t {...};</code>								
Values:	<table border="1"> <tr> <td>kSoftReset= 0</td> <td>softReset.</td> </tr> <tr> <td>kHardReset= 1</td> <td>hardReset</td> </tr> <tr> <td>kKeyOffOnReset= 2</td> <td>keyOffOnReset</td> </tr> <tr> <td>kCustomReset= 3</td> <td>kCustomReset</td> </tr> </table>	kSoftReset= 0	softReset.	kHardReset= 1	hardReset	kKeyOffOnReset= 2	keyOffOnReset	kCustomReset= 3	kCustomReset
kSoftReset= 0	softReset.								
kHardReset= 1	hardReset								
kKeyOffOnReset= 2	keyOffOnReset								
kCustomReset= 3	kCustomReset								
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>								
Description:	The type of the requested reset. @uptrace{}								

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.20.2 diag::EcuResetRequest::StopOffer function

[SWS_DM_01017]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::EcuResetRequest
Syntax:	<code>void StopOffer ();</code>
Return value:	None
Header file:	<code>#include "ara/diag/ecu_reset_request.h"</code>
Description:	This StopOffer will disable the forwarding of request messages from DM.

|(RS_Diag_04169, RS_Diag_04196)

8.3.1.20.3 diag::EcuResetRequest::Offer function

[SWS_DM_01016]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kNotOffered	There was no Offer called before.
	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/ecu_reset_request.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00139, RS_AP_00119)

8.3.1.20.4 diag::EcuResetRequest::RequestReset function

[SWS_DM_01013]{DRAFT} [

Kind:	function	
Symbol:	RequestReset(ResetRequestType resetType, ara::core::Optional< std::uint8_t > id, const Meta Info &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	virtual ara::core::Future<void> RequestReset (ResetRequestType reset Type, ara::core::Optional< std::uint8_t > id, const MetaInfo &meta Info, CancellationHandler cancellationHandler)=0;	
Parameters (in):	resetType	Type of the requested reset.
	id	id of the custom reset type. Will only be evaluated when resetType is "custom"
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< void >	–
Errors:	DiagErrorDomain::DiagErrc::kRejected	Requested operation was rejected due to State Managements/machines internal state.
	DiagErrorDomain::DiagErrc::kRequest Failed	Diagnostic request could not be performed successfully.
	DiagErrorDomain::DiagErrc::kCustom ResetTypeNotSupported	The requested Diagnostic custom reset type is not supported by the Diagnostic Address instance.
	DiagErrorDomain::DiagErrc::kReset TypeNotSupported	The requested Diagnostic reset type is not supported by the Diagnostic Address instance.





Header file:	#include "ara/diag/ecu_reset_request.h"
Description:	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](#))

8.3.1.20.5 diag::EcuResetRequest::ExecuteReset function

[SWS_DM_01014]{DRAFT} [

Kind:	function	
Symbol:	ExecuteReset(MetaInfo metaInfo)	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	virtual void ExecuteReset (MetaInfo metaInfo)=0;	
DIRECTION NOT DEFINED	metaInfo	-
Return value:	None	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Description:	StateManagement has to execute the requested reset. Called after DM sent the response message to tester.	

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.20.6 diag::EcuResetRequest::EnableRapidShutdown function

[SWS_DM_01012]{DRAFT} [

Kind:	function	
Symbol:	EnableRapidShutdown(bool enable, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	virtual ara::core::Future<void> EnableRapidShutdown (bool enable, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	enable	when enable is set to true the rapid shutdown will be enabled, setting enable to false will disable rapid shutdown
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< void >	-
Errors:	DiagErrorDomain::DiagErrc::kRejected	Requested operation was rejected due to State Managements/machines internal state.



△

	DiagErrorDomain::DiagErrc::kResetTypeNotSupported	The requested Diagnostic reset type is not supported by the Diagnostic Address instance.
Header file:	#include "ara/diag/ecu_reset_request.h"	
Description:	interface for subFunction En-/DisableRapidShutdown	

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.20.7 diag::EcuResetRequest::~EcuResetRequest destructor

[SWS_DM_01011]{DRAFT} [

Kind:	function	
Symbol:	~EcuResetRequest()	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	virtual ~EcuResetRequest () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/ecu_reset_request.h"	
Description:	Destructor of EcuResetRequest. @uptrace{}	

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.20.8 diag::EcuResetRequest::EcuResetRequest constructor

[SWS_DM_01010]{DRAFT} [

Kind:	function	
Symbol:	EcuResetRequest(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::EcuResetRequest	
Syntax:	explicit EcuResetRequest (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
Header file:	#include "ara/diag/ecu_reset_request.h"	
Description:	Constructor of EcuResetRequest. @uptrace{}	

]([RS_Diag_04169](#), [RS_Diag_04196](#))

8.3.1.21 Authentication class

[SWS_DM_01123]{DRAFT} [

Kind:	class
Symbol:	Authentication
Scope:	namespace ara::diag
Syntax:	<code>class Authentication final {...};</code>
Header file:	<code>#include "ara/diag/authentication.h"</code>
Description:	Class to implement the Service Authentication interfaces to application.

]([RS_Diag_04251](#))

8.3.1.21.1 diag::Authentication::Authentication constructor

[SWS_DM_01124]{DRAFT} [

Kind:	function	
Symbol:	Authentication(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::Authentication	
Syntax:	<code>explicit Authentication (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);</code>	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype of a Diagnostic Authentication service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	<code>#include "ara/diag/authentication.h"</code>	
Description:	Constructor of Authentication.	

]([RS_Diag_04251](#))

8.3.1.21.2 diag::Authentication::~~Authentication destructor

[SWS_DM_01125]{DRAFT} [

Kind:	function
Symbol:	~Authentication()
Scope:	class ara::diag::Authentication
Syntax:	<code>virtual ~Authentication () noexcept=default;</code>
Exception Safety:	noexcept
Header file:	<code>#include "ara/diag/authentication.h"</code>
Description:	Destructor of Authentication.

]([RS_Diag_04251](#))

8.3.1.21.3 diag::Authentication::VerifyCertificateUnidirectional function

[SWS_DM_01126]{DRAFT} [

Kind:	function	
Symbol:	VerifyCertificateUnidirectional(std::uint8_t CommunicationConfiguration, ara::core::Span< std::uint8_t > ClientCertificate, ara::core::Span< std::uint8_t > ClientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::Authentication	
Syntax:	virtual ara::core::Future<std::tuple<ara::core::Span<std::uint8_t>, ara::core::Span<std::uint8_t> > > VerifyCertificateUnidirectional (std::uint8_t CommunicationConfiguration, ara::core::Span< std::uint8_t > ClientCertificate, ara::core::Span< std::uint8_t > Client Challenge, const MetaInfo &metaInfo, CancellationHandler cancellation Handler)=0;	
Parameters (in):	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 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 Communication Configuration used
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	ara::core::Future< std::tuple< ara::core::Span< std::uint8_t >, ara::core::Span< std::uint8_t > > >	Challenge created by the application, Ephemeral Public Key of Server
Errors:	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidTimePeriod	Date and time of the server does not match the validity period of the Certificate.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidSignature	Signature of the Certificate could not be verified.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidChainOfTrust	Certificate could not be verified against stored information about the issuing authority.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidType	Certificate does not match the current requested use case.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidFormat	Certificate could not be evaluated because the format requirement has not been met.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidContent	Certificate could not be verified because the content does not match.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidScope	The scope of the Certificate does not match the contents of the server.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidCert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagErrorDomain::DiagReporting Errc::kChallengeCalculationFailed	The challenge could not be calculated on the server side.





Header file:	#include "ara/diag/authentication.h"
Description:	This function accepts the certificate received from the tester, verifies it, and creates a challenge and ephemeral public key that must be returned to the tester.

]([RS_Diag_04251](#))

8.3.1.21.4 diag::Authentication::VerifyCertificateBidirectional function

[SWS_DM_01127]{DRAFT} [

Kind:	function	
Symbol:	VerifyCertificateBidirectional(std::uint8_t CommunicationConfiguration, ara::core::Span<std::uint8_t> ClientCertificate, ara::core::Span<std::uint8_t> ClientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::Authentication	
Syntax:	virtual ara::core::Future<std::tuple<ara::core::Span<std::uint8_t>, ara::core::Span<std::uint8_t>, ara::core::Span<std::uint8_t>, ara::core::Span<std::uint8_t> > > VerifyCertificateBidirectional (std::uint8_t CommunicationConfiguration, ara::core::Span<std::uint8_t> ClientCertificate, ara::core::Span<std::uint8_t> ClientChallenge, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	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	Set if the current conversation is canceled.
Return value:	ara::core::Future< std::tuple< ara::core::Span< std::uint8_t >, ara::core::Span< std::uint8_t >, ara::core::Span< std::uint8_t >, ara::core::Span< std::uint8_t > > >	Challenge created by the application, Certificate of the server, ProofOfOwnership calculated by the server, Ephemeral Public Key of server
Errors:	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidTimePeriod	Date and time of the server does not match the validity period of the Certificate.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidSignature	Signature of the Certificate could not be verified.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidChainOfTrust	Certificate could not be verified against stored information about the issuing authority.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidType	Certificate does not match the current requested use case.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidFormat	Certificate could not be evaluated because the format requirement has not been met.





	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidContent	Certificate could not be verified because the content does not match.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidScope	The scope of the Certificate does not match the contents of the server.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidCert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagErrorDomain::DiagReporting Errc::kChallengeCalculationFailed	The challenge could not be calculated on the server side.
Header file:	#include "ara/diag/authentication.h"	
Description:	This function accepts the certificate and challenge received from the tester, verifies the certificate, and creates a challenge, Ephemeral Public Key and Proof Of Ownership that must be returned to the tester. The function also returns the server certificate that will be used by the tester to verify the Proof Of Ownership.	

](RS_Diag_04251)

8.3.1.21.5 diag::Authentication::VerifyOwnership function

[SWS_DM_01128]{DRAFT} [

Kind:	function	
Symbol:	VerifyOwnership(ara::core::Span< std::uint8_t > ClientPOWN, ara::core::Span< std::uint8_t > ClientEphemeralPublicKey, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::Authentication	
Syntax:	virtual ara::core::Span<std::uint8_t> VerifyOwnership (ara::core::Span< std::uint8_t > ClientPOWN, ara::core::Span< std::uint8_t > ClientEphemeralPublicKey, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	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	Set if the current conversation is canceled.
Return value:	ara::core::Span< std::uint8_t >	Session Key Info or error
Errors:	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidTimePeriod	Date and time of the server does not match the validity period of the Certificate.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidSignature	Signature of the Certificate could not be verified.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidChainOfTrust	Certificate could not be verified against stored information about the issuing authority.





	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidType	Certificate does not match the current requested use case.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidFormat	Certificate could not be evaluated because the format requirement has not been met.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidContent	Certificate could not be verified because the content does not match.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidScope	The scope of the Certificate does not match the contents of the server.
	DiagErrorDomain::DiagReporting Errc::kCertificateVerificationFailed InvalidCert	Certificate received from client is invalid, because the server has revoked access for some reason.
	DiagErrorDomain::DiagReporting Errc::kChallengeCalculationFailed	The challenge could not be calculated on the server side.
Header file:	#include "ara/diag/authentication.h"	
Description:	This function accepts the Proof Of Ownership received from the tester and verifies it with the Public Key of the certificate received in the verifycertificateunidirectional/ verifycertificatebidirectional against the server challenge created in the last call to verifycertificateunidirectional/ verifycertificatebidirectional.	

|(RS_Diag_04251)

8.3.1.21.6 diag::Authentication::TransmitCertificate function

[SWS_DM_01129]{DRAFT} [

Kind:	function	
Symbol:	TransmitCertificate(ara::core::Span< std::uint8_t > CertificateData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class ara::diag::Authentication	
Syntax:	virtual Ara::core::Result<void> TransmitCertificate (ara::core::Span< std::uint8_t > CertificateData, const MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	CertificateData	Certificate to be verified, transmitted by the tester
	metaInfo	MetaInfo of the request.
	cancellationHandler	Set if the current conversation is canceled.
Return value:	Ara::core::Result< void >	void
Errors:	DiagErrorDomain::DiagReporting Errc::kOwnershipVerificationFailed	Delivered Ownership does not match the provided challenge or could not verified with the own private key.
Header file:	#include "ara/diag/authentication.h"	
Description:	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)

8.3.1.21.7 diag::Authentication::Offer function

[SWS_DM_01130]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::Authentication	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/authentication.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

]([RS_Diag_04251](#))

8.3.1.21.8 diag::Authentication::StopOffer function

[SWS_DM_01131]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class ara::diag::Authentication	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/authentication.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

]([RS_Diag_04251](#))

8.3.1.22 ClientAuthentication class

[SWS_DM_01132]{DRAFT} [

Kind:	class	
Symbol:	ClientAuthentication	
Scope:	namespace ara::diag	
Syntax:	class ClientAuthentication final {...};	
Header file:	#include "ara/diag/client_authentication.h"	
Description:	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](#))

8.3.1.22.1 diag::ClientAuthentication::DiagnosticAuthState type

[SWS_DM_01133]{DRAFT} [

Kind:	enumeration	
Symbol:	DiagnosticAuthState	
Scope:	class ara::diag::ClientAuthentication	
Underlying type:	std::uint8_t	
Syntax:	enum class DiagnosticAuthState : std::uint8_t {...};	
Values:	kDeAuthenticated= 0x00	No Diagnostic Clients are currently authenticated.
	kAuthenticated= 0x01	A Diagnostic Client is currently authenticated.
Header file:	#include "ara/diag/client_authentication.h"	
Description:	Possible values of the Authentication State of the client.	

]([RS_Diag_04251](#))

8.3.1.22.2 diag::ClientAuthentication::DiagnosticAuthRole type

[SWS_DM_01134]{DRAFT} [

Kind:	type alias	
Symbol:	DiagnosticAuthRole	
Scope:	class ara::diag::ClientAuthentication	
Derived from:	ara::core::String	
Syntax:	using DiagnosticAuthRole = ara::core::String;	
Header file:	#include "ara/diag/client_authentication.h"	
Description:	The Supported values for the Diagnostic Authentication roles are specified in the Diagnostic Extract.	

]([RS_Diag_04251](#))

8.3.1.22.3 diag::ClientAuthentication::~~ClientAuthentication destructor

[SWS_DM_01136]{DRAFT} [

Kind:	function	
Symbol:	~ClientAuthentication()	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	virtual ~ClientAuthentication () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication.h"	
Description:	Destructor of ClientAuthentication.	

]([RS_Diag_04251](#))

8.3.1.22.4 diag::ClientAuthentication::ClientAuthentication copy constructor

[SWS_DM_01139]{DRAFT} [

Kind:	function
Symbol:	ClientAuthentication(ClientAuthentication const &other)
Scope:	class ara::diag::ClientAuthentication
Syntax:	ClientAuthentication (ClientAuthentication const &other)=delete;
Header file:	#include "ara/diag/client_authentication.h"
Description:	Copy constructor of ClientAuthentication cannot be used.

]([RS_Diag_04251](#))

8.3.1.22.5 diag::ClientAuthentication::ClientAuthentication move constructor

[SWS_DM_01137]{DRAFT} [

Kind:	function
Symbol:	ClientAuthentication(ClientAuthentication &&other)
Scope:	class ara::diag::ClientAuthentication
Syntax:	ClientAuthentication (ClientAuthentication &&other) noexcept=default;
Parameters (in):	other Object to move-construct from
Exception Safety:	noexcept
Header file:	#include "ara/diag/client_authentication.h"
Description:	Move constructor of ClientAuthentication.

]([RS_Diag_04251](#))

8.3.1.22.6 diag::ClientAuthentication::ClientAuthentication copy assignment operator

[SWS_DM_01140]{DRAFT} [

Kind:	function
Symbol:	operator=(ClientAuthentication const &other)
Scope:	class ara::diag::ClientAuthentication
Syntax:	auto operator= (ClientAuthentication const &other) -> Client Authentication &=delete;
Header file:	#include "ara/diag/client_authentication.h"
Description:	Copy assignment operator of ClientAuthentication cannot be used.

]([RS_Diag_04251](#))

8.3.1.22.7 diag::ClientAuthentication::ClientAuthentication move assignment operator

[SWS_DM_01138]{DRAFT} [

Kind:	function	
Symbol:	operator=(ClientAuthentication &&other)	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	auto operator= (ClientAuthentication &&other) &noexcept -> ClientAuthentication &=default;	
Parameters (in):	other	Object to move-assign from.
Return value:	ClientAuthentication &=default	–
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication.h"	
Description:	Move assignment operator of ClientAuthentication.	

](RS_Diag_04251)

8.3.1.22.8 diag::ClientAuthentication::OverrideDefaultRoles function

[SWS_DM_01141]{DRAFT} [

Kind:	function	
Symbol:	OverrideDefaultRoles(ara::core::Vector< DiagnosticAuthRole > defaultRoles, std::chrono::milliseconds timeout)	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	ara::core::Result<ClientAuthenticationHandle> OverrideDefaultRoles (ara::core::Vector< DiagnosticAuthRole > defaultRoles, std::chrono::milliseconds timeout);	
Parameters (in):	defaultRoles	The default roles requested by the application, to be set on the diagnostic client
	timeout	The timeout until which the override request is active
Return value:	ara::core::Result< ClientAuthentication Handle >	Operation result
Header file:	#include "ara/diag/client_authentication.h"	
Description:	This method is used by the application to temporarily change the default AuthenticationRole for a Diagnostic Server Instance. The diagnostic services allowed in the passed defaultRoles are now accessible to the tester for a time period defined in the parameter timeout.	

](RS_Diag_04251)

8.3.1.22.9 diag::ClientAuthentication::Authenticate function

[SWS_DM_01142]{DRAFT} [

Kind:	function	
Symbol:	Authenticate(ara::core::Vector< DiagnosticAuthRole > userRoles)	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	ara::core::Result<ClientAuthenticationHandle> Authenticate (ara::core::Vector< DiagnosticAuthRole > userRoles);	
Parameters (in):	userRoles	The user roles to set on the diagnostic client
Return value:	ara::core::Result< ClientAuthentication Handle >	A handler of the Authentication State, which can be used by the application to set or extend the Dynamic AccessList
Header file:	#include "ara/diag/client_authentication.h"	
Description:	This function is used by the application to report the authenticated state to the Diagnostic Server instance. The authentication could be either done using the Authentication interfaces of the Diagnostic Server instance, or through other means in the application.	

](RS_Diag_04251)

8.3.1.22.10 diag::ClientAuthentication::GetState function

[SWS_DM_01143]{DRAFT} [

Kind:	function	
Symbol:	GetState()	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	ara::core::Result<DiagnosticAuthState> GetState () const noexcept;	
Return value:	ara::core::Result< DiagnosticAuthState >	The Authentication State of the Diagnostic Client or error
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication.h"	
Description:	This function is used by the application to query the current authentication state of the diagnostic client.	

](RS_Diag_04251)

8.3.1.22.11 diag::ClientAuthentication::SetNotifier function

[SWS_DM_01144]{DRAFT} [

Kind:	function	
Symbol:	SetNotifier(std::function< void(DiagnosticAuthState)> notifier)	
Scope:	class ara::diag::ClientAuthentication	
Syntax:	ara::core::Result<void> SetNotifier (std::function< void(Diagnostic AuthState)> notifier) noexcept;	
Parameters (in):	notifier	The notifier to call on state transition
Return value:	ara::core::Result< void >	void





Exception Safety:	noexcept
Header file:	#include "ara/diag/client_authentication.h"
Description:	This function is used by the application to set a Notifier Function that shall be called by the Diagnostic Server instance when an Authentication Status Change Occurs. This may be used, for. E.g, to notify the application when a transition to kDeAuthenticated State occurred due to an S3 timeout.

]([RS_Diag_04251](#))

8.3.1.23 ClientAuthenticationHandle class

[SWS_DM_01145]{DRAFT} [

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

]([RS_Diag_04251](#))

8.3.1.23.1 diag::ClientAuthenticationHandle::ClientAuthenticationHandle constructor

[SWS_DM_01146]{DRAFT} [

Kind:	function
Symbol:	ClientAuthenticationHandle()
Scope:	class ara::diag::ClientAuthenticationHandle
Syntax:	<code>ClientAuthenticationHandle ();</code>
Header file:	#include "ara/diag/client_authentication_handle.h"
Description:	Constructor of ClientAuthenticationHandle.

]([RS_Diag_04251](#))

8.3.1.23.2 diag::ClientAuthenticationHandle::~~ClientAuthenticationHandle destructor

[SWS_DM_01147]{DRAFT} [

Kind:	function
Symbol:	~ClientAuthenticationHandle()
Scope:	class ara::diag::ClientAuthenticationHandle
Syntax:	virtual ~ClientAuthenticationHandle () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/client_authentication_handle.h"
Description:	Destructor of ClientAuthenticationHandle.

|(RS_Diag_04251)

8.3.1.23.3 diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy constructor

[SWS_DM_01150]{DRAFT} [

Kind:	function
Symbol:	ClientAuthenticationHandle(ClientAuthenticationHandle const &other)
Scope:	class ara::diag::ClientAuthenticationHandle
Syntax:	ClientAuthenticationHandle (ClientAuthenticationHandle const &other)=delete;
Header file:	#include "ara/diag/client_authentication_handle.h"
Description:	Copy constructor of ClientAuthenticationHandle cannot be used.

|(RS_Diag_04251)

8.3.1.23.4 diag::ClientAuthenticationHandle::ClientAuthenticationHandle move constructor

[SWS_DM_01148]{DRAFT} [

Kind:	function	
Symbol:	ClientAuthenticationHandle(ClientAuthenticationHandle &&other)	
Scope:	class ara::diag::ClientAuthenticationHandle	
Syntax:	ClientAuthenticationHandle (ClientAuthenticationHandle &&other) noexcept=default;	
Parameters (in):	other	Object to move-construct from
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Description:	Move constructor of ClientAuthenticationHandle.	

|(RS_Diag_04251)

8.3.1.23.5 diag::ClientAuthenticationHandle::ClientAuthenticationHandle copy assignment operator

[SWS_DM_01151]{DRAFT} [

Kind:	function
Symbol:	operator=(ClientAuthenticationHandle const &other)
Scope:	class ara::diag::ClientAuthenticationHandle
Syntax:	auto operator= (ClientAuthenticationHandle const &other) -> ClientAuthenticationHandle &=delete;
Header file:	#include "ara/diag/client_authentication_handle.h"
Description:	Copy assignment operator of CancellationHandler cannot be used.

]([RS_Diag_04251](#))

8.3.1.23.6 diag::ClientAuthenticationHandle::ClientAuthenticationHandle move assignment operator

[SWS_DM_01149]{DRAFT} [

Kind:	function	
Symbol:	operator=(ClientAuthenticationHandle &&other)	
Scope:	class ara::diag::ClientAuthenticationHandle	
Syntax:	auto operator= (ClientAuthenticationHandle &&other) &noexcept -> ClientAuthenticationHandle &=default;	
Parameters (in):	other	Object to move-assign from.
Return value:	ClientAuthenticationHandle &=default	-
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Description:	Move assignment operator of ClientAuthenticationHandle.	

]([RS_Diag_04251](#))

8.3.1.23.7 diag::ClientAuthenticationHandle::Append function

[SWS_DM_01152]{DRAFT} [

Kind:	function	
Symbol:	Append(DiagnosticServiceDynamicAccessList DynamicAccessList)	
Scope:	class ara::diag::ClientAuthenticationHandle	
Syntax:	ara::core::Result<void> Append (DiagnosticServiceDynamicAccessList DynamicAccessList) noexcept;	
Parameters (in):	DynamicAccessList	The DynamicAccessList to be appended in the client



△

Return value:	ara::core::Result< void >	void
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Description:	This function is used by the application to append a DynamicAccessList to the already existing DynamicAccessList of a Diagnostic Conversation.	

](RS_Diag_04251)

8.3.1.23.8 diag::ClientAuthenticationHandle::Set function

[SWS_DM_01153]{DRAFT} [

Kind:	function	
Symbol:	Set(DiagnosticServiceDynamicAccessList DynamicAccessList)	
Scope:	class ara::diag::ClientAuthenticationHandle	
Syntax:	ara::core::Result<void> Set (DiagnosticServiceDynamicAccessList DynamicAccessList) noexcept;	
Parameters (in):	DynamicAccessList	The new DynamicAccessList to be set in the client
Return value:	ara::core::Result< void >	void
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Description:	This function is used by the application to set/replace a DynamicAccessList of a diagnostic conversation.	

](RS_Diag_04251)

8.3.1.23.9 diag::ClientAuthenticationHandle::Revoke function

[SWS_DM_01154]{DRAFT} [

Kind:	function	
Symbol:	Revoke()	
Scope:	class ara::diag::ClientAuthenticationHandle	
Syntax:	ara::core::Result<void> Revoke () noexcept;	
Return value:	ara::core::Result< void >	void
Exception Safety:	noexcept	
Header file:	#include "ara/diag/client_authentication_handle.h"	
Description:	This function is used by the application to de-authenticate a client, and also to clear the DynamicAccessList and any overridden defaults.	

](RS_Diag_04251)

8.3.1.23.10 diag::ClientAuthenticationHandle::Refresh function

[SWS_DM_01155]{DRAFT} [

Kind:	function
Symbol:	Refresh()
Scope:	class ara::diag::ClientAuthenticationHandle
Syntax:	ara::core::Result<void> Refresh () noexcept;
Return value:	ara::core::Result< void > void
Exception Safety:	noexcept
Header file:	#include "ara/diag/client_authentication_handle.h"
Description:	This function is used by the application to refresh the timer that was started by Authenticate or OverrideDefaultRoles. If both Methods were previously called, both timers are refreshed.

]([RS_Diag_04251](#))

8.3.1.24 DiagnosticServiceDynamicAccessList class

[SWS_DM_01156]{DRAFT} [

Kind:	class
Symbol:	DiagnosticServiceDynamicAccessList
Scope:	namespace ara::diag
Syntax:	class DiagnosticServiceDynamicAccessList final {...};
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Description:	Definition of the DiagnosticServiceDynamicAccessList class, which is used by the application to build a DynamicAccessList.

]([RS_Diag_04251](#))

8.3.1.24.1 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList constructor

[SWS_DM_01157]{DRAFT} [

Kind:	function
Symbol:	DiagnosticServiceDynamicAccessList()
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList
Syntax:	DiagnosticServiceDynamicAccessList ();
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Description:	Constructor of DiagnosticServiceDynamicAccessList.

]([RS_Diag_04251](#))

8.3.1.24.2 diag::DiagnosticServiceDynamicAccessList::~~DiagnosticServiceDynamicAccessList destructor

[SWS_DM_01158]{DRAFT} [

Kind:	function
Symbol:	~DiagnosticServiceDynamicAccessList()
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList
Syntax:	virtual ~DiagnosticServiceDynamicAccessList () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Description:	Destructor of DiagnosticServiceDynamicAccessList.

]([RS_Diag_04251](#))

8.3.1.24.3 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy constructor

[SWS_DM_01159]{DRAFT} [

Kind:	function
Symbol:	DiagnosticServiceDynamicAccessList(DiagnosticServiceDynamicAccessList const &other)
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList
Syntax:	DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList const &other) noexcept;
Parameters (in):	other Object to copy-construct from
Exception Safety:	noexcept
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Description:	Copy constructor of DiagnosticServiceDynamicAccessList.

]([RS_Diag_04251](#))

8.3.1.24.4 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move constructor

[SWS_DM_01160]{DRAFT} [

Kind:	function
Symbol:	DiagnosticServiceDynamicAccessList(DiagnosticServiceDynamicAccessList &&other)
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList
Syntax:	DiagnosticServiceDynamicAccessList (DiagnosticServiceDynamicAccessList &&other) noexcept;
Parameters (in):	other Object to move-construct from



△

Exception Safety:	noexcept
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"
Description:	Move constructor of DiagnosticServiceDynamicAccessList.

]([RS_Diag_04251](#))

8.3.1.24.5 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList copy assignment operator

[SWS_DM_01161]{DRAFT} [

Kind:	function	
Symbol:	operator=(DiagnosticServiceDynamicAccessList const &other)	
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList	
Syntax:	auto operator= (DiagnosticServiceDynamicAccessList const &other) &-> DiagnosticServiceDynamicAccessList &;	
Parameters (in):	other	Object to copy-assign from.
Return value:	DiagnosticServiceDynamicAccessList &	–
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Description:	Copy assignment operator of DiagnosticServiceDynamicAccessList.	

]([RS_Diag_04251](#))

8.3.1.24.6 diag::DiagnosticServiceDynamicAccessList::DiagnosticServiceDynamicAccessList move assignment operator

[SWS_DM_01162]{DRAFT} [

Kind:	function	
Symbol:	operator=(DiagnosticServiceDynamicAccessList &&other)	
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList	
Syntax:	auto operator= (DiagnosticServiceDynamicAccessList &&other) &-> DiagnosticServiceDynamicAccessList &;	
Parameters (in):	other	Object to move-assign from.
Return value:	DiagnosticServiceDynamicAccessList &	–
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Description:	Move assignment operator of DiagnosticServiceDynamicAccessList.	

]([RS_Diag_04251](#))

8.3.1.24.7 diag::DiagnosticServiceDynamicAccessList::Reserve function

[SWS_DM_01163]{DRAFT} [

Kind:	function	
Symbol:	Reserve(std::size_t numberOfServiceHeads, std::size_t maxServiceHeadSize)	
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList	
Syntax:	void Reserve (std::size_t numberOfServiceHeads, std::size_t maxServiceHeadSize) noexcept;	
Parameters (in):	numberOfServiceHeads	The number of diagnostic service patterns
	maxServiceHeadSize	The expected maximum number of diagnostic service bytes in a single pattern
Return value:	None	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Description:	Preallocates memory for all service heads to fit into the DynamicAccessList. The preallocation can be just estimated and may calculate just the worst case of memory needed, not exact memory size needed for data numberOfServiceHeads.	

]([RS_Diag_04251](#))

8.3.1.24.8 diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function

[SWS_DM_01164]{DRAFT} [

Kind:	function	
Symbol:	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::Byte sid)	
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList	
Syntax:	auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte sid) noexcept -> DynamicAccessListDiagServiceBuilder;	
Parameters (in):	sid	The diagnostic service identifier
Return value:	DynamicAccessListDiagServiceBuilder	An instance of a diagnostic service pattern builder
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Description:	This function is used by the Application to construct a pattern for the DynamicAccessList using only the SID.	

]([RS_Diag_04251](#))

8.3.1.24.9 diag::DiagnosticServiceDynamicAccessList::MakeServiceBuilder function

[SWS_DM_01165]{DRAFT} [

Kind:	function	
Symbol:	MakeServiceBuilder(DynamicAccessListDiagServiceBuilder::ByteString serviceHead)	
Scope:	class ara::diag::DiagnosticServiceDynamicAccessList	
Syntax:	<pre>auto MakeServiceBuilder (DynamicAccessListDiagServiceBuilder::Byte String serviceHead) noexcept -> DynamicAccessListDiagServiceBuilder;</pre>	
Parameters (in):	serviceHead	A string of bytes to start the DynamicAccess pattern-match
Return value:	DynamicAccessListDiagServiceBuilder	An instance of a diagnostic service pattern builder
Exception Safety:	noexcept	
Header file:	#include "ara/diag/diagnostic_service_dynamic_access_list.h"	
Description:	This function is used by the Application to construct a pattern for the DynamicAccessList using a string of bytes.	

](RS_Diag_04251)

8.3.1.25 ExternalAuthentication class

[SWS_DM_01191]{DRAFT} [

Kind:	class	
Symbol:	ExternalAuthentication	
Scope:	namespace ara::diag	
Syntax:	<pre>class ExternalAuthentication final {...};</pre>	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	Definition of the ExternalAuthentication class, which is used by the application to receive an instance of the ClientAuthentication Class relevant to the specific client.	

](RS_Diag_04251)

8.3.1.25.1 diag::ExternalAuthentication::Address type

[SWS_DM_01192]{DRAFT} [

Kind:	type alias	
Symbol:	Address	
Scope:	class ara::diag::ExternalAuthentication	
Derived from:	std::uint16_t	
Syntax:	<pre>using Address = std::uint16_t;</pre>	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	Alias for tester address.	

](RS_Diag_04251)

8.3.1.25.2 diag::ExternalAuthentication::ExternalAuthentication constructor

[SWS_DM_01193]{DRAFT} [

Kind:	function	
Symbol:	ExternalAuthentication(ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	explicit ExternalAuthentication (ara::core::InstanceSpecifier instanceSpecifier, ReentrancyType reentrancyType) noexcept;	
Parameters (in):	instanceSpecifier	InstanceSpecifier to a PortPrototype of a Diagnostic Authentication service instance in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Exception Safety:	noexcept	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	Constructs the port for authentication of diagnostic clients.	

]([RS_Diag_04251](#))

8.3.1.25.3 diag::ExternalAuthentication::ExternalAuthentication copy constructor

[SWS_DM_01196]{DRAFT} [

Kind:	function	
Symbol:	ExternalAuthentication(ExternalAuthentication const &other)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	ExternalAuthentication (ExternalAuthentication const &other)=delete;	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	Copy constructor of ExternalAuthentication.	

]([RS_Diag_04251](#))

8.3.1.25.4 diag::ExternalAuthentication::ExternalAuthentication move constructor

[SWS_DM_01194]{DRAFT} [

Kind:	function	
Symbol:	ExternalAuthentication(ExternalAuthentication &&other)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	ExternalAuthentication (ExternalAuthentication &&other) noexcept=default;	
Parameters (in):	other	Object to move-construct from



△

Exception Safety:	noexcept
Header file:	#include "ara/diag/external_authentication.h"
Description:	Move constructor of ExternalAuthentication.

|(RS_Diag_04251)

8.3.1.25.5 diag::ExternalAuthentication::ExternalAuthentication copy assignment operator

[SWS_DM_01197]{DRAFT} [

Kind:	function
Symbol:	operator=(ExternalAuthentication const &other)
Scope:	class ara::diag::ExternalAuthentication
Syntax:	auto operator= (ExternalAuthentication const &other) -> ExternalAuthentication &=delete;
Header file:	#include "ara/diag/external_authentication.h"
Description:	Copy assignment operator of ExternalAuthentication.

|(RS_Diag_04251)

8.3.1.25.6 diag::ExternalAuthentication::ExternalAuthentication move assignment operator

[SWS_DM_01195]{DRAFT} [

Kind:	function	
Symbol:	operator=(ExternalAuthentication &&other)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	auto operator= (ExternalAuthentication &&other) &noexcept -> ExternalAuthentication &=default;	
Parameters (in):	other	Object to move-assign from.
Return value:	ExternalAuthentication &=default	-
Exception Safety:	noexcept	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	Move assignment operator of ExternalAuthentication.	

|(RS_Diag_04251)

8.3.1.25.7 diag::ExternalAuthentication::~~ExternalAuthentication destructor

[SWS_DM_01198]{DRAFT} [

Kind:	function
Symbol:	~ExternalAuthentication()
Scope:	class ara::diag::ExternalAuthentication
Syntax:	~ExternalAuthentication () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/external_authentication.h"
Description:	Destructor of DiagnosticServiceDynamicAccessList.

|(RS_Diag_04251)

8.3.1.25.8 diag::ExternalAuthentication::Get function

[SWS_DM_01199]{DRAFT} [

Kind:	function	
Symbol:	Get(ara::diag::MetaInfo &metaInfo)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	ara::core::Result<ClientAuthentication> Get (ara::diag::MetaInfo &metaInfo) noexcept;	
Parameters (in):	metaInfo	The meta information of a diagnostic service port
Return value:	ara::core::Result< ClientAuthentication >	The diagnostic client associated authentication object or error
Exception Safety:	noexcept	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	This function is used by the application to get the ClientAuthentication Instance that is handling the Authentication State of the Client corresponding to the MetaInfo.	

|(RS_Diag_04251)

8.3.1.25.9 diag::ExternalAuthentication::Get function

[SWS_DM_01200]{DRAFT} [

Kind:	function	
Symbol:	Get(Address sourceAddress)	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	ara::core::Result<ClientAuthentication> Get (Address sourceAddress) noexcept;	
Parameters (in):	sourceAddress	The source address of the client
Return value:	ara::core::Result< ClientAuthentication >	The associated diagnostic client authentication object or error
Exception Safety:	noexcept	
Header file:	#include "ara/diag/external_authentication.h"	





Description:	This function is used by the application to get the ClientAuthentication Instance that is handling the Authentication State of the Client corresponding to the Address.
---------------------	---

]([RS_Diag_04251](#))

8.3.1.25.10 diag::ExternalAuthentication::GetAll function

[SWS_DM_01201]{DRAFT} [

Kind:	function	
Symbol:	GetAll()	
Scope:	class ara::diag::ExternalAuthentication	
Syntax:	ara::core::Vector<ClientAuthentication> GetAll () noexcept;	
Return value:	ara::core::Vector< ClientAuthentication >	The list of all diagnostic client associated authentication objects or empty list if none available.
Exception Safety:	noexcept	
Header file:	#include "ara/diag/external_authentication.h"	
Description:	This function is used by the application to get all the ClientAuthentication Instances that are currently handled by the DM.	

]([RS_Diag_04251](#))

8.3.1.26 DynamicAccessListDiagServiceBuilder class

[SWS_DM_01166]{DRAFT} [

Kind:	class	
Symbol:	DynamicAccessListDiagServiceBuilder	
Scope:	namespace ara::diag	
Syntax:	class DynamicAccessListDiagServiceBuilder final {...};	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Definition of the DynamicAccessListDiagServiceBuilder class, which is used by the application to build a DynamicAccessList.	

]([RS_Diag_04251](#))

8.3.1.26.1 ByteRange class

[SWS_DM_01182]{DRAFT} [

Kind:	class
Symbol:	ByteRange
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder
Syntax:	<code>class ByteRange final {...};</code>
Header file:	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>
Description:	Represents a single byte value closed range. Type alias of a single diagnostic service pattern element in the DynamicAccessList

|(RS_Diag_04251)

8.3.1.26.1.1 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange constructor

[SWS_DM_01183]{DRAFT} [

Kind:	function	
Symbol:	ByteRange(Byte min, Byte max)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	<code>ByteRange (Byte min, Byte max) noexcept;</code>	
Parameters (in):	min	The minimum value to match
DIRECTION NOT DEFINED	max	–
Exception Safety:	noexcept	
Header file:	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>	
Description:	Constructs a value range.	

|(RS_Diag_04251)

8.3.1.26.1.2 diag::DynamicAccessListDiagServiceBuilder::~ByteRange::~~ByteRange destructor

[SWS_DM_01188]{DRAFT} [

Kind:	function	
Symbol:	~ByteRange()	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	<code>~ByteRange () noexcept;</code>	
Exception Safety:	noexcept	
Header file:	<code>#include "ara/diag/dynamic_access_list_diag_service_builder.h"</code>	
Description:	Destructor of ByteRange.	

|(RS_Diag_04251)

8.3.1.26.1.3 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy constructor

[SWS_DM_01184]{DRAFT} [

Kind:	function	
Symbol:	ByteRange(ByteRange const &other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	ByteRange (ByteRange const &other) noexcept;	
DIRECTION NOT DEFINED	other	–
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Copy constructor of ByteRange.	

]([RS_Diag_04251](#))

8.3.1.26.1.4 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move constructor

[SWS_DM_01185]{DRAFT} [

Kind:	function	
Symbol:	ByteRange(ByteRange &&other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	ByteRange (ByteRange &&other) noexcept;	
DIRECTION NOT DEFINED	other	–
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Move constructor of ByteRange.	

]([RS_Diag_04251](#))

8.3.1.26.1.5 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange copy assignment operator

[SWS_DM_01186]{DRAFT} [

Kind:	function	
Symbol:	operator=(ByteRange const &other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	auto operator= (ByteRange const &other) &-> ByteRange &;	



△

DIRECTION NOT DEFINED	other	–
Return value:	ByteRange &	–
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Copy assignment operator of ByteRange.	

](RS_Diag_04251)

8.3.1.26.1.6 diag::DynamicAccessListDiagServiceBuilder::ByteRange::ByteRange move assignment operator

[SWS_DM_01187]{DRAFT} [

Kind:	function	
Symbol:	operator=(ByteRange &&other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	auto operator= (ByteRange &&other) &-> ByteRange &;	
DIRECTION NOT DEFINED	other	–
Return value:	ByteRange &	–
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Move assignment operator of ByteRange.	

](RS_Diag_04251)

8.3.1.26.1.7 diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMin function

[SWS_DM_01189]{DRAFT} [

Kind:	function	
Symbol:	GetMin()	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	auto GetMin () const noexcept -> Byte;	
Return value:	Byte	The the lowest value
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Reports the lowest value to match.	

](RS_Diag_04251)

8.3.1.26.1.8 diag::DynamicAccessListDiagServiceBuilder::ByteRange::GetMax function

[SWS_DM_01190]{DRAFT} [

Kind:	function	
Symbol:	GetMax()	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder::ByteRange	
Syntax:	auto GetMax () const noexcept -> Byte;	
Return value:	Byte	The the highest value
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Reports the highest value to match.	

]([RS_Diag_04251](#))

8.3.1.26.2 diag::DynamicAccessListDiagServiceBuilder::Byte type

[SWS_DM_01167]{DRAFT} [

Kind:	type alias	
Symbol:	Byte	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Derived from:	std::uint8_t	
Syntax:	using Byte = std::uint8_t;	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Type alias of a single diagnostic service pattern element in the DynamicAccessList.	

]([RS_Diag_04251](#))

8.3.1.26.3 diag::DynamicAccessListDiagServiceBuilder::ByteString type

[SWS_DM_01168]{DRAFT} [

Kind:	type alias	
Symbol:	ByteString	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Derived from:	ara::core::Span<Byte>	
Syntax:	using ByteString = ara::core::Span<Byte>;	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Type alias of a sequence of diagnostic service pattern elements in the DynamicAccessList.	

]([RS_Diag_04251](#))

8.3.1.26.4 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessList-DiagServiceBuilder constructor

[SWS_DM_01169]{DRAFT} [

Kind:	function	
Symbol:	DynamicAccessListDiagServiceBuilder(V value, ara::core::Vector< std::uint8_t > &content)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	<pre>template <typename V> DynamicAccessListDiagServiceBuilder (V value, ara::core::Vector< std::uint8_t > &content) noexcept;</pre>	
Parameters (in):	value	value(s) to be added to the DynamicAccessList
	content	Stack holder for serialized DynamicAccessList
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Constructor for DynamicAccessListDiagServiceBuilder.	

](RS_Diag_04251)

8.3.1.26.5 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessList-DiagServiceBuilder copy constructor

[SWS_DM_01171]{DRAFT} [

Kind:	function	
Symbol:	DynamicAccessListDiagServiceBuilder(DynamicAccessListDiagServiceBuilder const &other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	<pre>DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagService Builder const &other)=delete;</pre>	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Copy constructor of DynamicAccessListDiagServiceBuilder cannot be used.	

](RS_Diag_04251)

8.3.1.26.6 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessList-DiagServiceBuilder move constructor

[SWS_DM_01170]{DRAFT} [

Kind:	function	
Symbol:	DynamicAccessListDiagServiceBuilder(DynamicAccessListDiagServiceBuilder &&other)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	





Syntax:	<code>DynamicAccessListDiagServiceBuilder (DynamicAccessListDiagServiceBuilder &&other) noexcept;</code>	
Parameters (in):	other	Object to move-construct from
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	Move constructor of DynamicAccessListDiagServiceBuilder.	

]([RS_Diag_04251](#))

8.3.1.26.7 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder copy assignment operator

[SWS_DM_01172]{DRAFT} [

Kind:	function
Symbol:	<code>operator=(DynamicAccessListDiagServiceBuilder const &other)</code>
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder
Syntax:	<code>auto operator= (DynamicAccessListDiagServiceBuilder const &other) -> DynamicAccessListDiagServiceBuilder &=delete;</code>
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Description:	Copy assignment operator of DynamicAccessListDiagServiceBuilder cannot be used.

]([RS_Diag_04251](#))

8.3.1.26.8 diag::DynamicAccessListDiagServiceBuilder::DynamicAccessListDiagServiceBuilder move assignment operator

[SWS_DM_01173]{DRAFT} [

Kind:	function
Symbol:	<code>operator=(DynamicAccessListDiagServiceBuilder &&other)</code>
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder
Syntax:	<code>auto operator= (DynamicAccessListDiagServiceBuilder &&other) -> DynamicAccessListDiagServiceBuilder &=delete;</code>
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Description:	Move assignment operator of DynamicAccessListDiagServiceBuilder.

]([RS_Diag_04251](#))

8.3.1.26.9 diag::DynamicAccessListDiagServiceBuilder::~~DynamicAccessListDiagServiceBuilder destructor

[SWS_DM_01174]{DRAFT} [

Kind:	function
Symbol:	~DynamicAccessListDiagServiceBuilder()
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder
Syntax:	~DynamicAccessListDiagServiceBuilder () noexcept;
Exception Safety:	noexcept
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Description:	Destructor of DynamicAccessListDiagServiceBuilder.

]([RS_Diag_04251](#))

8.3.1.26.10 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS_DM_01175]{DRAFT} [

Kind:	function	
Symbol:	Add(Byte value)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	auto Add (Byte value) noexcept -> DynamicAccessListDiagServiceBuilder &;	
Parameters (in):	value	The byte value to add to the DynamicAccessList. The value will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	This function is used by the application to add a single byte to a DynamicAccessListPattern.	

]([RS_Diag_04251](#))

8.3.1.26.11 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS_DM_01176]{DRAFT} [

Kind:	function
Symbol:	Add(ByteString values)
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder





Syntax:	auto Add (ByteString values) noexcept -> DynamicAccessListDiagServiceBuilder &;	
Parameters (in):	values	The byte sequence to add to the DynamicAccessList. The values will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	This function is used by the application to add a string of bytes to a DynamicAccessListPattern.	

]([RS_Diag_04251](#))

8.3.1.26.12 diag::DynamicAccessListDiagServiceBuilder::Add function

[SWS_DM_01177]{DRAFT} [

Kind:	function	
Symbol:	Add(ByteRange range)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	auto Add (ByteRange range) noexcept -> DynamicAccessListDiagServiceBuilder &;	
Parameters (in):	range	The range of byte values to add to the DynamicAccessList. The range will be used to check for a match during evaluation of the diagnostic service access rights
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	This function is used by the application to add a range of bytes to a DynamicAccessListPattern.	

]([RS_Diag_04251](#))

8.3.1.26.13 diag::DynamicAccessListDiagServiceBuilder::Any function

[SWS_DM_01178]{DRAFT} [

Kind:	function	
Symbol:	Any(std::size_t numberOfBytesToIgnore)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	auto Any (std::size_t numberOfBytesToIgnore) noexcept -> DynamicAccessListDiagServiceBuilder &;	





Parameters (in):	numberOfBytesToIgnore	The number of bytes to ignore
Return value:	DynamicAccessListDiagServiceBuilder &	The instance of the same object to allow fluent API usage
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	This function is used by the application to define a wildcard, i.e, to define a set of bytes that will be ignored in the DynamicAccessList pattern being created.	

](RS_Diag_04251)

8.3.1.26.14 diag::DynamicAccessListDiagServiceBuilder::EndsWith function

[SWS_DM_01179]{DRAFT} [

Kind:	function	
Symbol:	EndsWith(Byte value)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	void EndsWith (Byte value) noexcept;	
Parameters (in):	value	The byte value to end the DynamicAccessList. The value will be used to check for an exact match during evaluation of the diagnostic service access rights
Return value:	None	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	
Description:	This function is used by the application to specify a single byte value at the end of the Dynamic AccessList pattern.	

](RS_Diag_04251)

8.3.1.26.15 diag::DynamicAccessListDiagServiceBuilder::EndsWith function

[SWS_DM_01180]{DRAFT} [

Kind:	function	
Symbol:	EndsWith(ByteRange range)	
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder	
Syntax:	void EndsWith (ByteRange range) noexcept;	
Parameters (in):	range	The range of byte values to end the DynamicAccess List. The range will be used to check for a match during evaluation of the diagnostic service access rights
Return value:	None	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"	





Description:	This function is used by the application to specify a closed range of bytes at the end of the DynamicAccessList pattern.
---------------------	--

](RS_Diag_04251)

8.3.1.26.16 diag::DynamicAccessListDiagServiceBuilder::Build function

[SWS_DM_01181]{DRAFT} [

Kind:	function
Symbol:	Build()
Scope:	class ara::diag::DynamicAccessListDiagServiceBuilder
Syntax:	void Build () noexcept;
Return value:	None
Exception Safety:	noexcept
Header file:	#include "ara/diag/dynamic_access_list_diag_service_builder.h"
Description:	Finalizes building the DynamicAccessList. Must be called before destroying the ServiceBuilder object.

](RS_Diag_04251)

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} [

Kind:	class
Symbol:	ShortnameOfRIPortInterface





Scope:	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
Syntax:	class ShortnameOfRIPortInterface {...};
Header file:	#include "ara/diag/name_routine.h"
Description:	Typed Routine interface.

]([RS_Main_01002](#), [RS_Diag_04169](#))

8.3.2.2.1 Routine::StartOutput type

[SWS_DM_00581]{DRAFT} [

Kind:	struct
Symbol:	StartOutput
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
Syntax:	struct StartOutput {...};
Header file:	#include "ara/diag/name_routine.h"
Description:	Response data.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.2.2.2 Routine::StopOutput type

[SWS_DM_00582]{DRAFT} [

Kind:	struct
Symbol:	StopOutput
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
Syntax:	struct StopOutput {...};
Header file:	#include "ara/diag/name_routine.h"
Description:	Response data.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.2.2.3 Routine::RequestResultsOutput type

[SWS_DM_00583]{DRAFT} [

Kind:	struct
Symbol:	RequestResultsOutput





Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface
Syntax:	struct RequestResultsOutput {...};
Header file:	#include "ara/diag/name_routine.h"
Description:	Response data.

]([RS_Diag_04169](#), [RS_Diag_04224](#))

8.3.2.2.4 Routine constructor

[SWS_DM_00589]{DRAFT} [

Kind:	function	
Symbol:	ShortnameOfRIPortInterface(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	explicit ShortnameOfRIPortInterface (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/name_routine.h"	
Description:	Constructor of typed Routine interface.	

]([RS_AP_00137](#), [RS_Diag_04169](#))

8.3.2.2.5 Routine destructor

[SWS_DM_00590]{DRAFT} [

Kind:	function	
Symbol:	~ShortnameOfRIPortInterface()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	virtual ~ShortnameOfRIPortInterface () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/name_routine.h"	
Description:	Destructor of typed Routine interface.	

]([RS_AP_00134](#), [RS_Diag_04169](#))

8.3.2.2.6 Routine::Offer function

[SWS_DM_00594]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/name_routine.h"	
Description:	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](#))

8.3.2.2.7 Routine::StopOffer function

[SWS_DM_00595]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/name_routine.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

]([RS_Main_01002](#), [RS_Diag_04169](#))

8.3.2.2.8 Routine::Start function

[SWS_DM_00591]{DRAFT} [

Kind:	function	
Symbol:	Start(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentData Prototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... Meta Info &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	virtual ara::core::Future<StartOutput> Start (Namespace1OfTypeOf ArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1Of ArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2Of ArgumentDataPrototype Shortname2OfArgumentDataPrototype;... MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled





Return value:	ara::core::Future< StartOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.	
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.	
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.	





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/name_routine.h"	
Description:	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)

8.3.2.2.9 Routine::Stop function

[SWS_DM_00592]{DRAFT} [

Kind:	function
Symbol:	Stop(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... Meta Info &metaInfo, CancellationHandler cancellationHandler)
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface





Syntax:	virtual ara::core::Future<StopOutput> Stop (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype; Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< StopOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.	





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/name_routine.h"	
Description:	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)

8.3.2.2.10 Routine::RequestResults function

[SWS_DM_00593]{DRAFT} [

Kind:	function	
Symbol:	RequestResults(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... Meta Info &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfRIPortInterface	
Syntax:	virtual ara::core::Future<RequestResultsOutput> RequestResults (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< RequestResults Output >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.	



△

	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/name_routine.h"	
Description:	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 [Diagnostic-DataIdentifierInterface](#).

[SWS_DM_00601]{DRAFT} [

Kind:	class
Symbol:	ShortnameOfDIPortInterface
Scope:	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
Syntax:	<code>class ShortnameOfDIPortInterface {...};</code>
Header file:	<code>#include "ara/diag/name_data_identifier.h"</code>
Description:	Typed DataIdentifier interface.

]([RS_Diag_04169](#))

8.3.2.3.1 DataIdentifier::OperationOutput type

[SWS_DM_00579]{DRAFT} [

Kind:	struct
Symbol:	Output
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
Syntax:	<code>struct Output {...};</code>
Header file:	<code>#include "ara/diag/name_data_identifier.h"</code>
Description:	Response data.

]([RS_Diag_04169](#))

8.3.2.3.2 DataIdentifier constructor

[SWS_DM_00585]{DRAFT} [

Kind:	function	
Symbol:	ShortnameOfDIPortInterface(const ara::core::InstanceSpecifier &specifier, DataIdentifier ReentrancyType reentrancyType)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface	
Syntax:	<code>explicit ShortnameOfDIPortInterface (const ara::core::InstanceSpecifier &specifier, DataIdentifierReentrancyType reentrancyType);</code>	
Parameters (in):	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
Header file:	<code>#include "ara/diag/name_data_identifier.h"</code>	





Description:	Constructor of typed DataIdentifier interface.
---------------------	--

]([RS_AP_00137](#), [RS_Diag_04169](#))

8.3.2.3.3 DataIdentifier destructor

[SWS_DM_00586]{DRAFT} [

Kind:	function
Symbol:	~ShortnameOfDIPortInterface()
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
Syntax:	virtual ~ShortnameOfDIPortInterface () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/name_data_identifier.h"
Description:	Destructor of typed DataIdentifier interface.

]([RS_AP_00134](#), [RS_Diag_04169](#))

8.3.2.3.4 DataIdentifier::Offer function

[SWS_DM_00599]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/name_data_identifier.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

]([RS_AP_00119](#), [RS_AP_00139](#), [RS_Diag_04169](#))

8.3.2.3.5 DataIdentifier::StopOffer function

[SWS_DM_00600]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/name_data_identifier.h"
Description:	This StopOffer will disable the forwarding of request messages from DM.

|(RS_Diag_04169)

8.3.2.3.6 DataIdentifier::Read function

[SWS_DM_00640]{DRAFT} [

Kind:	function	
Symbol:	Read(MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface	
Syntax:	virtual ara::core::Future<Output> Read (MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< Output >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for a negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.	





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/name_data_identifier.h"	
Description:	Called for ReadDataByIdentifier request for this DiagnosticDataIdentifier.	

]([RS_AP_00119](#), [RS_AP_00138](#), [RS_Diag_04169](#), [RS_Diag_04170](#))

8.3.2.3.7 DataIdentifier::Write function

[SWS_DM_00598]{DRAFT} [

Kind:	function	
Symbol:	Write(Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDIPortInterface	
Syntax:	virtual ara::core::Future<void> Write (Namespace1OfTypeOfArgumentDataPrototype::Type1OfArgumentDataPrototype Shortname1OfArgumentDataPrototype;Namespace2OfTypeOfArgumentDataPrototype::Type2OfArgumentDataPrototype Shortname2OfArgumentDataPrototype;... MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< void >	a Result with either void (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralReject	0x10, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
Header file:	#include "ara/diag/name_data_identifier.h"	
Description:	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 [DiagnosticDataElementInterface](#).

[SWS_DM_00603]{DRAFT} [

Kind:	class
Symbol:	ShortnameOfDEPortInterface
Scope:	namespace Namespace1OfPortInterface::Namespace2OfPortInterface
Syntax:	class ShortnameOfDEPortInterface {...};
Header file:	#include "ara/diag/name_data_element.h"
Description:	Typed DataElement interface.

]([RS_Diag_04169](#))

8.3.2.4.1 DataElement::OperationOutput type

[SWS_DM_00580]{DRAFT} [

Kind:	struct
Symbol:	OperationOutput
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface
Syntax:	struct OperationOutput {...};



△

Header file:	#include "ara/diag/name_data_element.h"
Description:	Response data.

]([RS_Diag_04169](#))

8.3.2.4.2 DataElement constructor

[SWS_DM_00587]{DRAFT} [

Kind:	function	
Symbol:	ShortnameOfDEPortInterface(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface	
Syntax:	explicit ShortnameOfDEPortInterface (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	An InstanceSpecifier linking this instance with the PortPrototype in the manifest
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/name_data_element.h"	
Description:	Constructor of typed DataElement interface.	

]([RS_AP_00137](#), [RS_Diag_04169](#))

8.3.2.4.3 DataElement destructor

[SWS_DM_00588]{DRAFT} [

Kind:	function	
Symbol:	~ShortnameOfDEPortInterface()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface	
Syntax:	virtual ~ShortnameOfDEPortInterface () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/name_data_element.h"	
Description:	Destructor of typed DataElement interface.	

]([RS_AP_00134](#), [RS_Diag_04169](#))

8.3.2.4.4 DataElement::Offer function

[SWS_DM_00597]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/name_data_element.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00119, RS_AP_00139, RS_Diag_04169)

8.3.2.4.5 DataElement::StopOffer function

[SWS_DM_00617]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/name_data_element.h"	
Description:	This StopOffer will disable the forwarding of request messages from DM.	

|(RS_Diag_04169)

8.3.2.4.6 DataElement::Read function

[SWS_DM_00596]{DRAFT} [

Kind:	function	
Symbol:	Read(MetaInfo &metaInfo, CancellationHandler cancellationHandler)	
Scope:	class Namespace1OfPortInterface::Namespace2OfPortInterface::ShortnameOfDEPortInterface	
Syntax:	virtual ara::core::Future<OperationOutput> Read (MetaInfo &metaInfo, CancellationHandler cancellationHandler)=0;	
Parameters (in):	metaInfo	contains additional meta information
	cancellationHandler	informs if the current conversation is canceled
Return value:	ara::core::Future< OperationOutput >	a Result with either OperationOutput (for a positive response message) or an UDS NRC value (for an negative response message)
Errors:	DiagUdsNrcErrorDomain::DiagUdsNrc Errc::kGeneralReject	0x10, Negative return code according to ISO 14229.





DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupported	0x11, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubfunctionNotSupported	0x12, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kIncorrectMessageLengthOrInvalidFormat	0x13, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResponseTooLong	0x14, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBusyRepeatRequest	0x21, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kConditionsNotCorrect	0x22, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestSequenceError	0x24, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kNoResponseFromSubnetComponent	0x25, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kFailurePreventsExecutionOfRequestedAction	0x26, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequestOutOfRange	0x31, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSecurityAccessDenied	0x33, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kInvalidKey	0x35, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kExceedNumberOfAttempts	0x36, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRequiredTimeDelayNotExpired	0x37, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kUploadDownloadNotAccepted	0x70, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransferDataSuspended	0x71, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kGeneralProgrammingFailure	0x72, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kWrongBlockSequenceCounter	0x73, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kSubFunctionNotSupportedInActiveSession	0x7E, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kServiceNotSupportedInActiveSession	0x7F, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooHigh	0x81, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kRpmTooLow	0x82, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsRunning	0x83, Negative return code according to ISO 14229.
DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEnginelsNotRunning	0x84, Negative return code according to ISO 14229.





	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kEngineRunTimeTooLow	0x85, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooHigh	0x86, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTemperatureTooLow	0x87, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooHigh	0x88, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVehicleSpeedTooLow	0x89, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooHigh	0x8A, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kThrottlePedalTooLow	0x8B, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInNeutral	0x8C, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTransmissionRangeNotInGear	0x8D, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kBrakeSwitchNotClosed	0x8F, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kShifterLeverNotInPark	0x90, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kTorqueConverterClutchLocked	0x91, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooHigh	0x92, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kVoltageTooLow	0x93, Negative return code according to ISO 14229.
	DiagUdsNrcErrorDomain::DiagUdsNrcErrc::kResourceTemporarilyNotAvailable	0x94, Negative return code according to ISO 14229-1.
Header file:	#include "ara/diag/name_data_element.h"	
Description:	Called for reading a DataElement.	

|(RS_AP_00119, RS_AP_00138, RS_Diag_04170, RS_Diag_04169)

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.

Class	ApApplicationError
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface





Class	ApApplicationError			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform Tags: atp.Status=draft atp.recommendedPackage=ApplicationErrors			
Base	<i>ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
errorCode	Integer	1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error. Tags: atp.Status=draft
errorDomain	ApApplicationError Domain	1	ref	This reference represents the error domain of the Ap ApplicationError. Tags: atp.Status=draft

Table A.1: ApApplicationError

Class	CplusplusImplementationDataType (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CplusplusImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding Tags: atp.Status=draft			
Base	<i>ARElement, ARObject, AbstractImplementationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, CplusplusImplementationDataTypeContextTarget, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Subclasses	CustomCplusplusImplementationDataType, StdCplusplusImplementationDataType			
Attribute	Type	Mult.	Kind	Note
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CplusplusImplementationDataType has array semantics. Stereotypes: atpVariation Tags: atp.Status=draft vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration. Tags: atp.Status=draft
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CplusplusImplementationDataType. Tags: atp.Status=draft
subElement (ordered)	CplusplusImplementationDataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CplusplusImplementationDataType Tags: atp.Status=draft
template Argument (ordered)	CplusplusTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments Tags: atp.Status=draft





Class	CplusplusImplementationDataType (abstract)			
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CplusplusImplementationDataType is contributed to the language binding. Tags: atp.Status=draft
typeReference	CplusplusImplementationDataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). Tags: atp.Status=draft

Table A.2: CplusplusImplementationDataType

Class	DiagEventDebounceCounterBased			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the counter-based debounce algorithm shall be used by the DEM for this diagnostic monitor. This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounceCounterBased.			
Base	ARObject, DiagEventDebounceAlgorithm, Identifiable , MultilanguageReferrable , Referrable			
Attribute	Type	Mult.	Kind	Note
counterBasedFdcThresholdStorageValue	Integer	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame.
counterDecrementStepSize	Integer	0..1	attr	This value shall be taken to decrement the internal debounce counter. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterFailedThreshold	Integer	0..1	attr	This value defines the event-specific limit that indicates the "failed" counter status. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterIncrementStepSize	Integer	0..1	attr	This value shall be taken to increment the internal debounce counter. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterJumpDown	Boolean	0..1	attr	This value activates or deactivates the counter jump-down behavior. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterJumpDownValue	Integer	0..1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from incrementing to decrementing. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterJumpUp	Boolean	0..1	attr	This value activates or deactivates the counter jump-up behavior. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
counterJumpUpValue	Integer	0..1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from decrementing to incrementing. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime





Class		DiagEventDebounceCounterBased		
counterPassed Threshold	Integer	0..1	attr	This value defines the event-specific limit that indicates the "passed" counter status. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.3: DiagEventDebounceCounterBased

Class		DiagEventDebounceMonitorInternal		
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	"This meta-class represents the ability to indicate that no Dem pre-debounce algorithm shall be used for this diagnostic monitor. The SWC might implement an internal debouncing algorithm and report qualified (debounced) results to the Dem/DM.			
Base	ARObject, DiagEventDebounceAlgorithm, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.4: DiagEventDebounceMonitorInternal

Class		DiagEventDebounceTimeBased		
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the time-based pre-debounce algorithm shall be used by the Dem for this diagnostic monitor. This is related to set the EcuC choice container DemDebounceAlgorithmClass to DemDebounceTime Base.			
Base	ARObject, DiagEventDebounceAlgorithm, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i>			
Attribute	Type	Mult.	Kind	Note
timeBasedFdc Threshold StorageValue	TimeValue	0..1	attr	Threshold to allocate an event memory entry and to capture the Freeze Frame. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
timeFailed Threshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "failed" status. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
timePassed Threshold	TimeValue	0..1	attr	This value represents the event-specific delay indicating the "passed" status. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.5: DiagEventDebounceTimeBased

Class		DiagnosticAbstractDataIdentifier (abstract)		
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents an abstract base class for the modeling of a diagnostic data identifier (DID).			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i>			
Subclasses	DiagnosticDataIdentifier, DiagnosticDynamicDataIdentifier			
Attribute	Type	Mult.	Kind	Note





Class	DiagnosticAbstractDataIdentifier (abstract)			
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticAbstractDataIdentifier in the scope of diagnostic workflow Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.6: DiagnosticAbstractDataIdentifier

Class	DiagnosticAccessPermission			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This represents the specification of whether a given service can be accessed according to the existence of meta-classes referenced by a particular DiagnosticAccessPermission. In other words, this meta-class acts as a mapping element between several (otherwise unrelated) pieces of information that are put into context for the purpose of checking for access rights. Tags: atp.recommendedPackage=DiagnosticAccessPermissions			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
authentication Role	DiagnosticAuthRole	*	ref	This reference identifies the authenticationRole applicable for the enclosing DiagnosticMemoryDestinationUser Defined.
diagnostic Session	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions
environmental Condition	Diagnostic EnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission.
securityLevel	DiagnosticSecurityLevel	*	ref	This represents the associated DiagnosticSecurityLevels

Table A.7: DiagnosticAccessPermission

Class	DiagnosticAging			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticAging			
Note	Defines the aging algorithm. Tags: atp.recommendedPackage=DiagnosticAgings			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
agingCycle	DiagnosticOperation Cycle	0..1	ref	This represents the applicable aging cycle. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=agingCycle.diagnosticOperationCycle, aging Cycle.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
threshold	PositiveInteger	0..1	attr	Number of aging cycles needed to unlearn/delete the event. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.8: DiagnosticAging

Class	DiagnosticAuthentication (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::Authentication			
Note	This meta-class represents the ability to configure the usage of the UDS service Authentication in the Diagnostic extract.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Subclasses	DiagnosticAuthenticationConfiguration, DiagnosticDeAuthentication, DiagnosticProofOfOwnership, DiagnosticVerifyCertificateBidirectional, DiagnosticVerifyCertificateUnidirectional			
Attribute	Type	Mult.	Kind	Note
authentication Class	Diagnostic AuthenticationClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference.
authentication Timeout	TimeValue	0..1	attr	This attribute defines the time that the authentication state is maintained in default-session if there is no communication from the authenticated client.

Table A.9: DiagnosticAuthentication

Class	DiagnosticClearCondition			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticConditions			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticCondition, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.10: DiagnosticClearCondition

Enumeration	DiagnosticClearDtcLimitationEnum			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	Scope of the DEM_ClearDTC Api.			
Literal	Description			
allSupportedDtcs	DEM_ClearDtc API accepts all supported DTC values. Tags: atp.EnumerationLiteralIndex=0			
clearAllDtcs	DEM_ClearDtc API accepts ClearAllDTCs only. Tags: atp.EnumerationLiteralIndex=1			

Table A.11: DiagnosticClearDtcLimitationEnum

Enumeration	DiagnosticClearEventAllowedBehaviorEnum			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This enumeration defines the possible behavior for clear event allowed			





Enumeration	DiagnosticClearEventAllowedBehaviorEnum
Literal	Description
noStatusByteChange	The event status byte keeps unchanged. Tags: atp.EnumerationLiteralIndex=0
onlyThisCycleAndReadiness	The OperationCycle and readiness bits of the event status byte are reset. Tags: atp.EnumerationLiteralIndex=1

Table A.12: DiagnosticClearEventAllowedBehaviorEnum

Class	DiagnosticComControl			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl			
Note	This represents an instance of the "Communication Control" diagnostic service. Tags: atp.recommendedPackage=DiagnosticCommunicationControls			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
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.13: DiagnosticComControl

Class	<<atpVariation>> DiagnosticCommonProps			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps			
Note	This meta-class aggregates a number of common properties that are shared among a diagnostic extract. Tags: vh.latestBindingTime=codeGenerationTime			
Base	<i>ARObject</i>			
Attribute	Type	Mult.	Kind	Note
debounceAlgorithmProps	DiagnosticDebounceAlgorithmProps	*	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..*.
defaultEndianness	ByteOrderEnum	0..1	attr	Defines the default endianness of the data belonging to a DID or RID which is applicable if the DiagnosticDataElement does not define the endianness via the swDataDefProps.baseType attribute.
eventCombinationReportingBehavior	DiagnosticEventCombinationReportingBehaviorEnum	0..1	attr	In case of EventCombination on Retrieval, this attribute specifies if a specific order of reporting is to be maintained.





Class	<<atpVariation>> DiagnosticCommonProps			
maxNumberOfRequestCorrectlyReceivedResponsePending	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.
occurrenceCounterProcessing	DiagnosticOccurrenceCounterProcessingEnum	0..1	attr	This attribute defines the consideration of the fault confirmation process for the occurrence counter.
resetConfirmedBitOnOverflow	Boolean	0..1	attr	This attribute defines, whether the confirmed bit is reset or not while an event memory entry will be displaced.
responseOnAllRequestSids	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).
responseOnSecondDeclinedRequest	Boolean	0..1	attr	Defines the reaction upon a second request (ClientB) that can not be processed (e.g. due to priority assessment). TRUE: when the second request (Client B) can not be processed, it shall be answered with NRC21 BusyRepeat Request. FALSE: when the second request (Client B) can not be processed, it shall not be responded.
typeOfEventCombinationSupported	DiagnosticEventCombinationBehaviorEnum	0..1	attr	Select type of Event Combination support.

Table A.14: DiagnosticCommonProps

Class	DiagnosticCommunicationManagerNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	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.			
Base	<i>ARObject</i> , <i>DiagnosticCapabilityElement</i> , Identifiable , <i>MultilanguageReferrable</i> , <i>Referrable</i> , <i>ServiceNeeds</i>			
Attribute	Type	Mult.	Kind	Note
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.15: DiagnosticCommunicationManagerNeeds

Class	DiagnosticConditionInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for diagnostic conditions on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	<i>ARElement</i> , <i>ARObject</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>CollectableElement</i> , DiagnosticPortInterface , Identifiable , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , PortInterface , <i>Referrable</i>			





Class		DiagnosticConditionInterface		
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

Table A.16: DiagnosticConditionInterface

Class		DiagnosticConnectedIndicator		
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	Description of indicators that are defined per DiagnosticEvent.			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable			
Attribute	Type	Mult.	Kind	Note
behavior	DiagnosticConnectedIndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycle	DiagnosticOperationCycle	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 Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
indicator	DiagnosticIndicator	0..1	ref	Reference to the used indicator.

Table A.17: DiagnosticConnectedIndicator

Class		DiagnosticContributionSet		
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticContribution			
Note	This meta-class represents a root node of a diagnostic extract. It bundles a given set of diagnostic model elements. The granularity of the DiagnosticContributionSet is arbitrary in order to support the aspect of decentralized configuration, i.e. different contributors can come up with an own DiagnosticContributionSet. Tags: atp.recommendedPackage=DiagnosticContributionSets			
Base	ARElement, ARObject, CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
commonProperties	DiagnosticCommonProps	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire DiagnosticContributionSet. Stereotypes: atpSplittable Tags: atp.Splitkey=commonProperties
element	DiagnosticCommonElement	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=element.diagnosticCommonElement, element.variationPoint.shortLabel, vh.latestBindingTime=postBuild





Class	DiagnosticContributionSet			
serviceTable	DiagnosticServiceTable	*	ref	<p>This represents the collection of DiagnosticServiceTables to be considered in the scope of this DiagnosticContributionSet.</p> <p>Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=serviceTable.diagnosticServiceTable, serviceTable.variationPoint.shortLabel vh.latestBindingTime=postBuild</p>

Table A.18: DiagnosticContributionSet

Class	DiagnosticCustomServiceInstance			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance			
Note	<p>This meta-class has the ability to define an instance of a custom diagnostic service.</p> <p>Tags:atp.recommendedPackage=DiagnosticCustomInstances</p>			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
customServiceClass	DiagnosticCustomServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustomServiceClass.

Table A.19: DiagnosticCustomServiceInstance

Class	DiagnosticDTCInformationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	<p>This meta-class represents the ability to implement a PortInterface to access the properties of DTCs on the adaptive platform.</p> <p>Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces</p>			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.20: DiagnosticDTCInformationInterface

Class	DiagnosticDataByIdentifier (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an abstract base class for all diagnostic services that access data by identifier.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Subclasses	DiagnosticReadDataByIdentifier , DiagnosticReadScalingDataByIdentifier , DiagnosticWriteDataByIdentifier			
Attribute	Type	Mult.	Kind	Note
dataIdentifier	DiagnosticAbstractDataIdentifier	0..1	ref	This represents the linked DiagnosticDataIdentifier.

Table A.21: DiagnosticDataByIdentifier

Class	DiagnosticDataElement			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe a concrete piece of data to be taken into account for diagnostic purposes.			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable			
Attribute	Type	Mult.	Kind	Note
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOfElements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticDataElement if used with DiagnosticReadScalingDataByIdentifier
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints.

Table A.22: DiagnosticDataElement

Class	DiagnosticDataElementInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a element-of-DID-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractDataIdentifierInterface, DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of an element of a diagnostic data identifier. Tags: atp.Status=draft

Table A.23: DiagnosticDataElementInterface

Class	DiagnosticDataIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to model a diagnostic data identifier (DID) that is fully specified regarding the payload at configuration-time. Tags: atp.recommendedPackage=DiagnosticDataIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticAbstractDataIdentifier , DiagnosticCommonElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the DiagnosticDataIdentifier. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=dataElement.bitOffset, dataElement.variation Point.shortLabel vh.latestBindingTime=postBuild





Class	DiagnosticDataIdentifier			
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the Diagnostic DataIdentifier.
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfo Byte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table A.24: DiagnosticDataIdentifier

Class	DiagnosticDataIdentifierGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic DID-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractDataIdentifierInterface, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.25: DiagnosticDataIdentifierGenericInterface

Class	DiagnosticDataIdentifierInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a DID-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractDataIdentifierInterface, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
read	ClientServerOperation	0..1	aggr	This represents the method to read the content of a diagnostic data identifier. Tags: atp.Status=draft
write	ClientServerOperation	0..1	aggr	This represents the method to write the contents of a diagnostic data identifier. Tags: atp.Status=draft

Table A.26: DiagnosticDataIdentifierInterface

Class	DiagnosticDataIdentifierSet			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This represents the ability to define a list of DiagnosticDataIdentifiers that can be reused in different contexts. Tags: atp.recommendedPackage=DiagnosticDataIdentifierSets			





Class	DiagnosticDataIdentifierSet			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
dataIdentifier (ordered)	DiagnosticDataIdentifier	*	ref	Reference to an ordered list of Data Identifiers.

Table A.27: DiagnosticDataIdentifierSet

Class	DiagnosticDataPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticServiceMappings			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, DiagnosticSwMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
diagnosticDataElement	DiagnosticDataElement	0..1	ref	This reference represents the applicable DiagnosticDataElement. Tags: atp.Status=draft
diagnosticDataIdentifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticDataIdentifier. Tags: atp.Status=draft
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype from which that data is obtained. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). Stereotypes: atpUriDef Tags: atp.Status=draft InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
process	ProcessDesign	1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. Stereotypes: atpSplittable Tags: atp.Splitkey=process atp.Status=draft

Table A.28: DiagnosticDataPortMapping

Class	DiagnosticDebounceAlgorithmProps			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticDebouncingAlgorithm			
Note	Defines properties for the debounce algorithm class.			
Base	<i>ARObject, Identifiable, MultilanguageReferrable, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
debounceAlgorithm	DiagEventDebounceAlgorithm	0..1	aggr	This represents the actual debounce algorithm.





Class	DiagnosticDebounceAlgorithmProps			
debounce Behavior	DiagnosticDebounceBehaviorEnum	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. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.29: DiagnosticDebounceAlgorithmProps

Enumeration	DiagnosticDebounceBehaviorEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticDebouncingAlgorithm
Note	Event debounce algorithm behavior options.
Literal	Description
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). Tags: 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. Tags: atp.EnumerationLiteralIndex=1

Table A.30: DiagnosticDebounceBehaviorEnum

Class	DiagnosticDoIPActivationLineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.31: DiagnosticDoIPActivationLineInterface

Class	DiagnosticDoIPGroupIdentificationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–





Class	DiagnosticDoIPGroupIdentificationInterface			
–	–	–	–	–

Table A.32: DiagnosticDoIPGroupIdentificationInterface

Class	DiagnosticDoIPPowerModelInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Power Mode on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.33: DiagnosticDoIPPowerModelInterface

Class	DiagnosticDoIPTriggerVehicleAnnouncementInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.34: DiagnosticDoIPTriggerVehicleAnnouncementInterface

Class	DiagnosticDownloadInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for downloading data using diagnostic channels on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.35: DiagnosticDownloadInterface

Class	DiagnosticDynamicallyDefineDataIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
Note	This represents an instance of the "Dynamically Define Data Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
dataIdentifier	DiagnosticDynamicData Identifier	0..1	ref	This represents the applicable DiagnosticDynamicData Identifier.
dynamically DefineData IdentifierClass	DiagnosticDynamically DefineDataIdentifier Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticDynamicallyDefineDataIdentifier 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.36: DiagnosticDynamicallyDefineDataIdentifier

Class	DiagnosticDynamicallyDefineDataIdentifierClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DynamicallyDefineDataIdentifier			
Note	This meta-class contains attributes shared by all instances of the "Dynamically Define Data Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDynamicallyDefineDataIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
checkPer SourceId	Boolean	0..1	attr	If set to TRUE, the Dcm module shall check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0x F200 to 0xF3FF. If set to FALSE, the Dcm module shall not check the session, security and mode dependencies per source DIDs with a ReadDataByIdentifier (0x22) with DID in the range 0xF200 to 0xF3FF.
configuration Handling	DiagnosticHandleDDDI ConfigurationEnum	0..1	attr	This configuration switch defines whether DDDID definition is handled as non-volatile information or not.
subfunction	DiagnosticDynamically DefineDataIdentifier SubfunctionEnum	*	attr	This attribute contains a list of applicable subfunctions for all DiagnosticDynamicallyDefineDataIdentifier that reference the DiagnosticDynamicallyDefineDataIdentifier Class.

Table A.37: DiagnosticDynamicallyDefineDataIdentifierClass

Class	DiagnosticEcuReset			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
Note	This represents an instance of the "ECU Reset" diagnostic service. Tags: atp.recommendedPackage=DiagnosticEcuResets			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note





Class		DiagnosticEcuReset		
customSubFunctionNumber	PositiveInteger	0..1	attr	This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used.
ecuResetClass	DiagnosticEcuResetClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticEcuReset in the given context.

Table A.38: DiagnosticEcuReset

Class		DiagnosticEcuResetClass		
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
Note	This meta-class contains attributes shared by all instances of the "Ecu Reset" diagnostic service. Tags: atp.recommendedPackage=DiagnosticEcuResets			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
respondToReset	DiagnosticResponseToEcuResetEnum	0..1	attr	This attribute defines whether the response to the Ecu Reset service shall be transmitted before or after the actual reset.

Table A.39: DiagnosticEcuResetClass

Class		DiagnosticEnableCondition		
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Specification of an enable condition. Tags: atp.recommendedPackage=DiagnosticConditions			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticCondition, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.40: DiagnosticEnableCondition

Class		<i>DiagnosticEnvCompareCondition</i> (abstract)		
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EnvironmentalCondition			
Note	DiagnosticCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in DiagnosticCompareCondition.compareType.			
Base	<i>ARObject, DiagnosticEnvConditionFormulaPart</i>			
Subclasses	DiagnosticEnvDataCondition , DiagnosticEnvModeCondition			
Attribute	Type	Mult.	Kind	Note
compareType	DiagnosticCompareTypeEnum	1	attr	This attributes represents the concrete type of the comparison.

Table A.41: DiagnosticEnvCompareCondition

Class	DiagnosticEnvConditionFormula			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EnvironmentalCondition			
Note	<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>			
Base	ARObject, DiagnosticEnvConditionFormulaPart			
Attribute	Type	Mult.	Kind	Note
nrcValue	PositiveInteger	0..1	attr	This attribute represents the concrete NRC value that shall be returned if the condition fails.
op	DiagnosticLogicalOperatorEnum	0..1	attr	This attribute represents the concrete operator (supported operators: and, or) of the condition formula.
part (ordered)	DiagnosticEnvConditionFormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

Table A.42: DiagnosticEnvConditionFormula

Class	DiagnosticEnvConditionFormulaPart (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EnvironmentalCondition			
Note	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompareCondition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
Base	ARObject			
Subclasses	DiagnosticEnvCompareCondition , DiagnosticEnvConditionFormula			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.43: DiagnosticEnvConditionFormulaPart

Class	DiagnosticEnvDataCondition			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EnvironmentalCondition			
Note	A DiagnosticEnvDataCondition is an atomic condition that compares the current value of the referenced DiagnosticDataElement with a constant value defined by the ValueSpecification. All compareTypes are supported.			
Base	ARObject, DiagnosticEnvCompareCondition , DiagnosticEnvConditionFormulaPart			
Attribute	Type	Mult.	Kind	Note
compareValue	ValueSpecification	0..1	aggr	This attribute represents a fixed compare value taken to evaluate the compare condition.
dataElement	DiagnosticDataElement	0..1	ref	This reference represents the related diagnostic data element.

Table A.44: DiagnosticEnvDataCondition

Class	DiagnosticEnvironmentalCondition			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EnvironmentalCondition			





Class	DiagnosticEnvironmentalCondition			
Note	The meta-class DiagnosticEnvironmentalCondition formalizes the idea of a condition which is evaluated during runtime of the ECU by looking at "environmental" states (e.g. one such condition is that the vehicle is not driving, i.e. vehicle speed == 0). Tags: atp.recommendedPackage=DiagnosticEnvironmentalConditions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
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.45: DiagnosticEnvironmentalCondition

Class	DiagnosticEvent			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	This element is used to configure DiagnosticEvents. Tags: atp.recommendedPackage=DiagnosticEvents			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
associated Event Identification	PositiveInteger	0..1	attr	This attribute represents the identification number that is associated with the enclosing DiagnosticEvent and allows to identify it when placed into a snapshot record or extended data record storage. This value can be reported as internal data element in snapshot records or extended data records.
clearEvent Allowed Behavior	DiagnosticClearEvent AllowedBehaviorEnum	0..1	attr	This attribute defines the resulting UDS status byte for the related event, which shall not be cleared according to the ClearEventAllowed callback
confirmation Threshold	PositiveInteger	0..1	attr	This attribute defines the number of operation cycles with a failed result before a confirmed DTC is set to 1. The semantic of this attribute is a by "1" increased value compared to the confirmation threshold of the "trip counter" mentioned in ISO 14229-1 in figure D.4. A value of "1" defines the immediate confirmation of the DTC along with the first reported failed. This is also sometimes called "zero trip DTC". A value of "2" defines a DTC confirmation in the operation cycle after the first occurred failed. A value of "2" is typically used in the US for OBD DTC confirmation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=connectedIndicator.shortName, connectedIndicator.variationPoint.shortLabel vh.latestBindingTime=postBuild
eventClear Allowed	DiagnosticEventClear AllowedEnum	0..1	attr	This attribute defines whether the Dem has access to a "ClearEventAllowed" callback.





<i>Class</i>	DiagnosticEvent			
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 ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)
recoverableIn SameOperation Cycle	Boolean	0..1	attr	If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting PASSED will be ignored and not lead to a reset of the indication of a failed test.

Table A.46: DiagnosticEvent

<i>Enumeration</i>	DiagnosticEventCombinationBehaviorEnum
<i>Package</i>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
<i>Note</i>	Select type of Event Combination support
<i>Literal</i>	Description
eventCombination OnRetrieval	Event combination on retrieval is used to combine events. For each event an individual event memory entry is created, while reporting the data via UDS, the data is combined. Tags: atp.EnumerationLiteralIndex=1
eventCombination OnStorage	Event combination on storage is used to combine events. Only one memory entry exists for each DTC which is also reported via UDS. Tags: atp.EnumerationLiteralIndex=0

Table A.47: DiagnosticEventCombinationBehaviorEnum

<i>Enumeration</i>	DiagnosticEventCombinationReportingBehaviorEnum
<i>Package</i>	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps
<i>Note</i>	Select reporting format of events. Applicable only for Event Combination on Retrieval.
<i>Literal</i>	Description
reportingIn ChronologicalOrder OldestFirst	The reporting order for event combination on retrieval is the chronological storage order of the events Tags: atp.EnumerationLiteralIndex=0

Table A.48: DiagnosticEventCombinationReportingBehaviorEnum

<i>Enumeration</i>	DiagnosticEventDisplacementStrategyEnum
<i>Package</i>	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
<i>Note</i>	Defines the displacement strategy.
<i>Literal</i>	Description
full	Event memory entry displacement is enabled, by consideration of priority active/passive status, and occurrence. Tags: atp.EnumerationLiteralIndex=0





Enumeration	DiagnosticEventDisplacementStrategyEnum
none	Event memory entry displacement is disabled. Tags: atp.EnumerationLiteralIndex=1
prioOcc	Event memory entry displacement is enabled, by consideration of priority and occurrence (but without active/passive status). Tags: atp.EnumerationLiteralIndex=2

Table A.49: DiagnosticEventDisplacementStrategyEnum

Class	DiagnosticEventInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to access the properties of diagnostic events on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.50: DiagnosticEventInterface

Class	DiagnosticEventToDebounceAlgorithmMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which Debounce Algorithm is applicable for a DiagnosticEvent. Tags: atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
debounceAlgorithm	DiagnosticDebounceAlgorithmProps	0..1	ref	Reference to a DebounceAlgorithm assigned to a DiagnosticEvent.
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which a Debounce Algorithm is assigned.

Table A.51: DiagnosticEventToDebounceAlgorithmMapping

Class	DiagnosticEventToEnableConditionGroupMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which EnableConditionGroup is applicable for a DiagnosticEvent. Tags: atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which an Enable ConditionGroup is assigned.





Class	DiagnosticEventToEnableConditionGroupMapping			
enableConditionGroup	DiagnosticEnableConditionGroup	0..1	ref	Reference to an EnableConditionGroup assigned to a DiagnosticEvent.

Table A.52: DiagnosticEventToEnableConditionGroupMapping

Class	DiagnosticEventToOperationCycleMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which OperationCycle is applicable for a DiagnosticEvent. Tags: atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which an Operation Cycle is assigned.
operationCycle	DiagnosticOperationCycle	0..1	ref	Reference to an OperationCycle assigned to a Diagnostic Event.

Table A.53: DiagnosticEventToOperationCycleMapping

Class	DiagnosticEventToTroubleCodeUdsMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines which UDS Diagnostic Trouble Code is applicable for a DiagnosticEvent. Tags: atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to a DiagnosticEvent to which a UDS Diagnostic Trouble Code is assigned.
troubleCodeUds	DiagnosticTroubleCodeUds	0..1	ref	Reference to an UDS Diagnostic Trouble Code assigned to a DiagnosticEvent.

Table A.54: DiagnosticEventToTroubleCodeUdsMapping

Class	DiagnosticEventWindow			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
Note	This represents the ability to define the characteristics of the applicable event window			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
eventWindowTime	DiagnosticEventWindowTimeEnum	0..1	attr	This attribute clarifies the validity of the eventWindow

Table A.55: DiagnosticEventWindow

Class	DiagnosticExtendedDataRecord			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticExtendedDataRecord			





Class	DiagnosticExtendedDataRecord			
Note	Description of an extended data record. Tags: atp.recommendedPackage=DiagnosticExtendedDataRecords			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
recordElement	DiagnosticParameter	*	aggr	Defined DataElements in the extended record element.
recordNumber	PositiveInteger	0..1	attr	This attribute specifies an unique identifier for an extended data record.
trigger	DiagnosticRecord TriggerEnum	0..1	attr	This attribute specifies the primary trigger to allocate an event memory entry.
update	Boolean	0..1	attr	This attribute defines when an extended data record is captured. True: This extended data record is captured every time. False: This extended data record is only captured for new event memory entries.

Table A.56: DiagnosticExtendedDataRecord

Class	DiagnosticFreezeFrame			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticFreezeFrame			
Note	This element describes combinations of DIDs for a non OBD relevant freeze frame. Tags: atp.recommendedPackage=DiagnosticFreezeFrames			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
customTrigger	String	0..1	attr	This attribute shall be taken to verbally describe the nature of the custom trigger.
recordNumber	PositiveInteger	0..1	attr	This attribute defines a record number for a freeze frame record. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
trigger	DiagnosticRecord TriggerEnum	0..1	attr	This attribute defines the primary trigger to allocate an event memory entry.
update	Boolean	0..1	attr	This attribute defines the approach when the freeze frame record is stored/updated. True: FreezeFrame record is captured every time. False: FreezeFrame record is only captured for new event memory entries.

Table A.57: DiagnosticFreezeFrame

Class	DiagnosticGenericUdsInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			





Class	DiagnosticGenericUdsInterface			
Note	This meta-class represents the ability to implement a generic UDS PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.58: DiagnosticGenericUdsInterface

Class	DiagnosticIndicatorInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement indicator functionality on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.59: DiagnosticIndicatorInterface

Class	DiagnosticMapping (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Abstract element for different kinds of diagnostic mappings.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Subclasses	DiagnosticEventToDebounceAlgorithmMapping , DiagnosticEventToEnableConditionGroupMapping , DiagnosticEventToOperationCycleMapping , DiagnosticEventToTroubleCodeUdsMapping , DiagnosticFimAliasEventGroupMapping , DiagnosticFimAliasEventMapping , DiagnosticInhibitSourceEventMapping , DiagnosticProvidedDataMapping , DiagnosticServiceDataMapping , DiagnosticSwMapping , DiagnosticTroubleCodeUdsToClearConditionGroupMapping , DiagnosticTroubleCodeUdsToTroubleCodeObdMapping			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.60: DiagnosticMapping

Class	DiagnosticMemoryDestination (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Subclasses	DiagnosticMemoryDestinationPrimary , DiagnosticMemoryDestinationUserDefined			





Class		DiagnosticMemoryDestination (abstract)			
Attribute	Type	Mult.	Kind	Note	
agingRequiresTestedCycle	Boolean	0..1	attr	Defines whether the aging cycle counter is processed every aging cycles or else only tested aging cycle are considered. If the attribute is set to TRUE: only tested aging cycle are considered for aging cycle counter. If the attribute is set to FALSE: aging cycle counter is processed every aging cycle.	
clearDtcLimitation	DiagnosticClearDtcLimitationEnum	0..1	attr	Defines the scope of the DEM_ClearDTC Api.	
dtcStatusAvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.	
eventDisplacementStrategy	DiagnosticEventDisplacementStrategyEnum	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	DiagnosticMemoryEntryStorageTriggerEnum	0..1	attr	Describes the trigger to allocate an event memory entry.	
statusBitHandlingTestFailedSinceLastClear	DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum	0..1	attr	This attribute defines, whether the aging and displacement mechanism shall be applied to the "TestFailedSinceLastClear" status bits.	
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	
typeOfFreezeFrameRecordNumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.	

Table A.61: DiagnosticMemoryDestination

Class		DiagnosticMemoryDestinationPortMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping				
Note	Defines to which SWC service ports with DiagnosticsEventInfoNeeds the DiagnosticMemoryDestination is mapped. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticMappings				
Base	ARElement , ARObject , CollectableElement , DiagnosticCommonElement , DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable				
Attribute	Type	Mult.	Kind	Note	
memoryDestination	DiagnosticMemoryDestination	1	ref	Reference to the MemoryDestination which is mapped to a SWC service port with DiagnosticEventInfoNeeds. Tags: atp.Status=draft	





Class		DiagnosticMemoryDestinationPortMapping		
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. Stereotypes: atpSplittable Tags: atp.Splitkey=process atp.Status=draft
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MemoryDestinationMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef Tags: atp.Status=draft InstanceRef implemented by: RPortPrototypeInExecutableInstanceRef

Table A.62: DiagnosticMemoryDestinationPortMapping

Class		DiagnosticMemoryDestinationPrimary		
Package		M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode		
Note		This represents a primary memory for a diagnostic event. Tags: atp.recommendedPackage=DiagnosticMemoryDestinations		
Base		ARElement , ARObject , CollectableElement , DiagnosticCommonElement , DiagnosticMemoryDestination , Identifiable , MultilanguageReferrable , PackageableElement , Referrable		
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.63: DiagnosticMemoryDestinationPrimary

Class		DiagnosticMemoryDestinationUserDefined		
Package		M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode		
Note		This represents a user-defined memory for a diagnostic event. Tags: atp.recommendedPackage=DiagnosticMemoryDestinations		
Base		ARElement , ARObject , CollectableElement , DiagnosticCommonElement , DiagnosticMemoryDestination , Identifiable , MultilanguageReferrable , PackageableElement , Referrable		
Attribute	Type	Mult.	Kind	Note
authenticationRole	DiagnosticAuthRole	0..1	ref	This reference identifies the authenticationRole applicable for the enclosing DiagnosticMemoryDestinationUserDefined.
memoryId	PositiveInteger	0..1	attr	This represents the identifier of the user-defined memory.

Table A.64: DiagnosticMemoryDestinationUserDefined

Class		DiagnosticMonitorInterface		
Package		M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface		





Class	DiagnosticMonitorInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.65: DiagnosticMonitorInterface

Enumeration	DiagnosticOccurrenceCounterProcessingEnum			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps			
Note	The occurrence counter triggering types.			
Literal	Description			
confirmedDtcBit	The occurrence counter is triggered by the TestFailed bit if the fault confirmation was successful (ConfirmedDTC bit is set). Tags: atp.EnumerationLiteralIndex=0			
testFailedBit	The occurrence counter is only triggered by the TestFailed bit (and the fault confirmation is not considered). Tags: atp.EnumerationLiteralIndex=1			

Table A.66: DiagnosticOccurrenceCounterProcessingEnum

Class	DiagnosticOperationCycle			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle			
Note	Definition of an operation cycle that is the base of the event qualifying and for Dem scheduling. Tags: atp.recommendedPackage=DiagnosticOperationCycles			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
cycleStatusStorage	Boolean	0..1	attr	Defines if the operation cycle state is available over the power cycle (stored non-volatile) or not. <ul style="list-style-type: none"> • true: the operation cycle state is stored non-volatile • false: the operation cycle state is only stored volatile This attribute is only relevant for the AUTOSAR adaptive platform. It no longer has a meaning on the AUTOSAR classic platform.
type	DiagnosticOperationCycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table A.67: DiagnosticOperationCycle

Class	DiagnosticOperationCycleInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for operation cycles on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
-	-	-	-	-

Table A.68: DiagnosticOperationCycleInterface

Class	DiagnosticParameter			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe information relevant for the execution of a specific diagnostic service, i.e. it can be taken to parameterize the service.			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
bitOffset	PositiveInteger	0..1	attr	This represents the bitOffset of the DiagnosticParameter Stereotypes: atpIdentityContributor
dataElement	DiagnosticDataElement	0..1	aggr	This represents the related dataElement of the DiagnosticParameter Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=dataElement.shortName, dataElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
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.69: DiagnosticParameter

Class	DiagnosticPeriodicRate			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
Note	This represents the ability to define a periodic rate for the specification of the "read data by periodic ID" diagnostic service.			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
period	TimeValue	0..1	attr	This represents the period of the DiagnosticPeriodicRate in seconds.
periodicRateCategory	DiagnosticPeriodicRateCategoryEnum	0..1	attr	This attribute represents the category of the periodic rate.

Table A.70: DiagnosticPeriodicRate

Class	DiagnosticPortInterface (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces. Tags: atp.Status=draft			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface , Referrable			
Subclasses	DiagnosticAbstractDataIdentifierInterface , DiagnosticAbstractRoutineInterface , DiagnosticAuthenticationInterface , DiagnosticComControllInterface , DiagnosticConditionInterface , DiagnosticDTCInformationInterface , DiagnosticDoIPActivationLineInterface , DiagnosticDoIPGroupIdentificationInterface , DiagnosticDoIPPowerModelInterface , DiagnosticDoIPTriggerVehicleAnnouncementInterface , DiagnosticDownloadInterface , DiagnosticEcuResetInterface , DiagnosticEventInterface , DiagnosticExternalAuthenticationInterface , DiagnosticGenericUdsInterface , DiagnosticIndicatorInterface , DiagnosticMonitorInterface , DiagnosticOperationCycleInterface , DiagnosticSecurityLevelInterface , DiagnosticServiceValidationInterface , DiagnosticUploadInterface			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.71: DiagnosticPortInterface

Class	DiagnosticProvidedDataMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
Note	This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly. Tags: atp.Status=draft atp.recommendedPackage=DataMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedData Mapping.dataProvider. Tags: atp.Status=draft
dataProvider	NameToken	1	attr	This represents the ability to further specify the data provider. Tags: atp.Status=draft

Table A.72: DiagnosticProvidedDataMapping

Class	DiagnosticReadDTCInformation			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDTCInformation			
Note	This represents an instance of the "Read DTC Information" diagnostic service. Tags: atp.recommendedPackage=DiagnosticReadDtcInformations			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
read DTCInformation Class	DiagnosticReadDTC InformationClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRead DTCInformation in the given context.

Table A.73: DiagnosticReadDTCInformation

Class	DiagnosticReadDataByIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an instance of the "Read Data by Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticDataByIdentifier , DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
readClass	DiagnosticReadDataByIdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticReadDataBy Identifier in the given context.

Table A.74: DiagnosticReadDataByIdentifier

Class	DiagnosticReadDataByIdentifierClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This meta-class contains attributes shared by all instances of the "Read Data by Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDataByIdentifiers			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
maxDidToRead	PositiveInteger	0..1	attr	This attribute represents the maximum number of allowed DIDs in a single instance of DiagnosticReadDataBy Identifier.

Table A.75: DiagnosticReadDataByIdentifierClass

Class	DiagnosticReadDataByPeriodicIDClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ReadDataByPeriodicID			
Note	This meta-class contains attributes shared by all instances of the "Read Data by periodic Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticReadDataByPeriodicIds			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
maxPeriodicDid ToRead	PositiveInteger	0..1	attr	This represents the maximum number of data identifiers that can be included in one request.
periodicRate	DiagnosticPeriodicRate	*	aggr	This represents the description of a collection of periodic rates in which the service can be executed.
schedulerMax Number	PositiveInteger	0..1	attr	This represents the maximum number of periodic data identifiers that can be scheduled in parallel.

Table A.76: DiagnosticReadDataByPeriodicIDClass

Class	DiagnosticResponseOnEvent			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent			
Note	This represents an instance of the "Response on Event" diagnostic service. Tags: atp.recommendedPackage=DiagnosticResponseOnEvents			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance , Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
eventWindow	DiagnosticEventWindow	*	aggr	This represents the applicable DiagnosticEventWindows
responseOnEventAction	DiagnosticResponseOnEventActionEnum	0..1	attr	Defines sub-functions of the service ResponseOnEvent.
responseOnEventClass	DiagnosticResponseOnEventClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticResponseOnEvent in the given context.

Table A.77: DiagnosticResponseOnEvent

Enumeration	DiagnosticResponseOnEventActionEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent
Note	This meta-class has the ability to define sub-functions of the UDS service ResponseOnEvent.
Literal	Description
clear	Clears the configured events. Tags: atp.EnumerationLiteralIndex=2
onChangeOfDataIdentifier	Reports based on change of data identifier. Tags: atp.EnumerationLiteralIndex=6
onComparisonOfValues	Triggered if data condition is met (e.g. RPM over 5000 1/min). Tags: atp.EnumerationLiteralIndex=8
onDTCStatusChange	Reports based on change of DTC status. Tags: atp.EnumerationLiteralIndex=7
report	Reports the activated events. Tags: atp.EnumerationLiteralIndex=3
reportDTCRecordInformationOnDtcStatusChange	Reports the DTC record-related information based on a DTC status change. (Subfunction 0x09) Tags: atp.EnumerationLiteralIndex=5
reportMostRecentDtcOnStatusChange	Triggers the report of the most recent failed or confirmed DTC (Subfunction 0x08). Tags: atp.EnumerationLiteralIndex=4
start	Starts the response on event service. Tags: atp.EnumerationLiteralIndex=1
stop	Stops the response on event service. Tags: atp.EnumerationLiteralIndex=0

Table A.78: DiagnosticResponseOnEventActionEnum

Class	DiagnosticResponseOnEventClass
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::ResponseOnEvent





Class	DiagnosticResponseOnEventClass			
Note	This represents the ability to define common properties for all instances of the "Response on Event" diagnostic service. Tags: atp.recommendedPackage=DiagnosticResponseOnEvents			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
maxNumberOfStoredDTCStatusChangedEvents	PositiveInteger	0..1	attr	The maximum number of DTCs that can be stored as DTCs with change status within one ResponseOnEvent SchedulerRate interval.
maxNumChangeOfDataIdentifierEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function onChangeOfDataIdentifier.
maxNumComparisonOfValueEvents	PositiveInteger	0..1	attr	The maximum number of events that can be simultaneously configured with sub function onComparisonOfValues.
maxSupportedDIDLength	PositiveInteger	0..1	attr	The maximum number of measurable data bytes allowed for each DID that is used for comparison or data change.
responseOnEventSchedulerRate	TimeValue	0..1	attr	The call rate of the periodic scheduler to compare the values of the DataIdentifier (DID) or to detect DTC status changes.

Table A.79: DiagnosticResponseOnEventClass

Enumeration	DiagnosticResponseToEcuResetEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset
Note	This enumeration controls the point in time in which a response to the reception of an EcuReset service shall be generated.
Literal	Description
respondAfterReset	Answer to EcuReset service should come after the reset. Tags: atp.EnumerationLiteralIndex=0
respondBeforeReset	Answer to EcuReset service should come before the reset. Tags: atp.EnumerationLiteralIndex=1

Table A.80: DiagnosticResponseToEcuResetEnum

Class	DiagnosticRoutine			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to define a diagnostic routine. Tags: atp.recommendedPackage=DiagnosticRoutines			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticRoutine in the scope of diagnostic workflow Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
requestResult	DiagnosticRequestRoutineResults	0..1	aggr	This represents the ability to request the result of a running routine.





Class	DiagnosticRoutine			
routineInfo	PositiveInteger	0..1	attr	This represents the routine info byte. The info byte contains a manufacturer-specific value (for the identification of record identifiers) that is reported to the tester. Other use cases for this attribute are mentioned in ISO 27145 and ISO 26021.
start	DiagnosticStartRoutine	0..1	aggr	This represents the ability to start a routine
stop	DiagnosticStopRoutine	0..1	aggr	This represents the ability to stop a running routine.

Table A.81: DiagnosticRoutine

Class	DiagnosticRoutineGenericInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a generic Routine-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractRoutineInterface, DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.82: DiagnosticRoutineGenericInterface

Class	DiagnosticRoutineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a routine-focused PortInterface for diagnostics on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticAbstractRoutineInterface, DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
requestResult	ClientServerOperation	0..1	aggr	This represents the request result method of the diagnostic routine. Tags: atp.Status=draft
start	ClientServerOperation	0..1	aggr	This represents the start method of the diagnostic routine. Tags: atp.Status=draft
stop	ClientServerOperation	0..1	aggr	This represents the stop method of the diagnostic routine. Tags: atp.Status=draft

Table A.83: DiagnosticRoutineInterface

Class	DiagnosticRoutineSubfunction (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class acts as an abstract base class to routine subfunctions.			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable			
Subclasses	DiagnosticRequestRoutineResults, DiagnosticStartRoutine, DiagnosticStopRoutine			
Attribute	Type	Mult.	Kind	Note
access Permission	DiagnosticAccess Permission	0..1	ref	This reference represents the access permission of the owning routine subfunction.

Table A.84: DiagnosticRoutineSubfunction

Class	DiagnosticSecurityAccess			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
Note	This represents an instance of the "Security Access" diagnostic service. Tags: atp.recommendedPackage=DiagnosticSecurityAccess			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceInstance , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
requestSeedId	PositiveInteger	0..1	attr	This would be 0x01, 0x03, 0x05, ... The sendKey id can be computed by adding 1 to the requestSeedId
securityAccess Class	DiagnosticSecurity AccessClass	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 at ECU boot power-on time where the Dcm remains in the default session and does not accept a security access.
securityLevel	DiagnosticSecurityLevel	0..1	ref	This reference identifies the applicable security level for the security access. Stereotypes: atpSplitable Tags: atp.Splitkey=securityLevel

Table A.85: DiagnosticSecurityAccess

Class	DiagnosticSecurityAccessClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. Tags: atp.recommendedPackage=DiagnosticSecurityAccess			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticServiceClass , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note





Class	DiagnosticSecurityAccessClass			
sharedTimer	Boolean	0..1	attr	<p>Switch between separate or single shared timer instance and timer value.</p> <ul style="list-style-type: none"> • True: use shared timer instance and timer value for all security access levels combined. • False: use separate timer instance and timer values for each security level. <p>Tags:atp.Status=draft</p>

Table A.86: DiagnosticSecurityAccessClass

Class	DiagnosticSecurityLevel			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	<p>This meta-class represents the ability to define a security level considered for diagnostic purposes.</p> <p>Tags:atp.recommendedPackage=DiagnosticSecurityLevels</p>			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
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.87: DiagnosticSecurityLevel

Class	DiagnosticSecurityLevelInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	<p>This meta-class represents the ability to implement a security-level-focused PortInterface for diagnostics on the adaptive platform.</p> <p>Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces</p>			
Base	<i>ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.88: DiagnosticSecurityLevelInterface

Class	DiagnosticServiceClass (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	<p>This meta-class provides the ability to define common properties that are shared among all instances of sub-classes of DiagnosticServiceInstance.</p>			





Class	DiagnosticServiceClass (abstract)			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Subclasses	DiagnosticAuthenticationClass, DiagnosticClearDiagnosticInformationClass, DiagnosticClearResetEmissionRelatedInfoClass, DiagnosticComControlClass, DiagnosticControlDTCSettingClass, DiagnosticCustomServiceClass, DiagnosticDataTransferClass, DiagnosticDynamicallyDefineDataIdentifierClass , DiagnosticEcuResetClass , DiagnosticIoControlClass, DiagnosticReadDTCInformationClass, DiagnosticReadDataByIdentifierClass , DiagnosticReadDataByPeriodicIDClass , DiagnosticReadMemoryByAddressClass, DiagnosticReadScalingDataByIdentifierClass, DiagnosticRequestControlOfOnBoardDeviceClass, DiagnosticRequestCurrentPowertrainDataClass, DiagnosticRequestDownloadClass, DiagnosticRequestEmissionRelatedDTCClass, DiagnosticRequestEmissionRelatedDTCPermanentStatusClass, DiagnosticRequestFileTransferClass, DiagnosticRequestOnBoardMonitoringTestResultsClass, DiagnosticRequestPowertrainFreezeFrameDataClass, DiagnosticRequestUploadClass, DiagnosticRequestVehicleInfoClass, DiagnosticResponseOnEventClass , DiagnosticRoutineControlClass, DiagnosticSecurityAccessClass , DiagnosticSessionControlClass, DiagnosticTransferExitClass, DiagnosticWriteDataByIdentifierClass, DiagnosticWriteMemoryByAddressClass			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.89: DiagnosticServiceClass

Class	DiagnosticServiceGenericMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticServiceMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , DiagnosticSwMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
diagnosticServiceInstance	DiagnosticServiceInstance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port. Tags: atp.Status=draft
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the DiagnosticServiceGenericMapping on the AUTOSAR adaptive platform. Stereotypes: atpUriDef Tags: atp.Status=draft InstanceRef implemented by: PPortPrototypeInExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. Stereotypes: atpSplitable Tags: atp.Splitkey=process atp.Status=draft

Table A.90: DiagnosticServiceGenericMapping

Class	DiagnosticServiceInstance (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents a concrete instance of a diagnostic service.			





Class	DiagnosticServiceInstance (abstract)			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Subclasses	DiagnosticAuthentication, DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelated Info, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, DiagnosticDataByIdentifier, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, Diagnostic IOControl, DiagnosticMemoryByAddress, DiagnosticReadDTCInformation, DiagnosticReadDataBy PeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequest PowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, Diagnostic RoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Attribute	Type	Mult.	Kind	Note
access Permission	DiagnosticAccess Permission	0..1	ref	This represents the collection of DiagnosticAccess Permissions that allow for the execution of the referencing DiagnosticServiceInstance..
serviceClass	DiagnosticServiceClass	0..1	ref	This represents the corresponding "class", i.e. this meta-class provides properties that are shared among all instances of applicable sub-classes of DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. Stereotypes: atpAbstract

Table A.91: DiagnosticServiceInstance

Class	DiagnosticServiceValidationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for service validation on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.92: DiagnosticServiceValidationInterface

Class	DiagnosticSession			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a diagnostic session. Tags: atp.recommendedPackage=DiagnosticSessions			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note
id	PositiveInteger	0..1	attr	This is the numerical identifier used to identify the DiagnosticSession in the scope of diagnostic workflow





Class	DiagnosticSession			
p2ServerMax	TimeValue	0..1	attr	This is the session value for P2ServerMax in seconds (per Session Control). The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds.
p2StarServerMax	TimeValue	0..1	attr	This is the session value for P2*ServerMax in seconds (per Session Control). The AUTOSAR configuration standard is to use SI units, so this parameter is defined as a float value in seconds.

Table A.93: DiagnosticSession

Enumeration	DiagnosticSignificanceEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	Significance level of a diagnostic event.
Literal	Description
fault	Failure, which affects the component/ECU itself. Tags: atp.EnumerationLiteralIndex=0
occurrence	Issue, which indicates additional information concerning insufficient system behavior. Tags: atp.EnumerationLiteralIndex=1

Table A.94: DiagnosticSignificanceEnum

Enumeration	DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.
Literal	Description
statusBitAgingAndDisplacement	The "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies. Tags: atp.EnumerationLiteralIndex=0
statusBitNormal	Aging and displacement has no impact on the "TestFailedSinceLastClear" status bits. Tags: atp.EnumerationLiteralIndex=1

Table A.95: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum

Class	DiagnosticTroubleCodeGroup			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	The diagnostic trouble code group defines the DTCs belonging together and thereby forming a group. Tags: atp.recommendedPackage=DiagnosticTroubleCodes			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Type	Mult.	Kind	Note





Class		DiagnosticTroubleCodeGroup		
dtc	DiagnosticTroubleCode	*	ref	This represents the collection of DiagnosticTroubleCodes defined by this DiagnosticTroubleCodeGroup. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=dtc.diagnosticTroubleCode, dtc.variationPoint.shortLabel vh.latestBindingTime=postBuild
groupNumber	PositiveInteger	0..1	attr	This represents the base number of the DTC group. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.96: DiagnosticTroubleCodeGroup

Class		DiagnosticTroubleCodeProps		
Package		M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode		
Note		This element defines common Dtc properties that can be reused by different non OBD-relevant DTCs. Tags: atp.recommendedPackage=DiagnosticTroubleCodePropss		
Base		<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable</i>		
Attribute	Type	Mult.	Kind	Note
aging	DiagnosticAging	0..1	ref	Reference to an aging algorithm in case that an aging/unlearning of the event is allowed.
diagnostic Memory	DiagnosticMemory Destination	0..1	ref	Reference to the applicable DiagnosticMemory Destination.
extendedData Record	DiagnosticExtended DataRecord	*	ref	Defines the links to an extended data class sampler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=extendedDataRecord.diagnosticExtendedDataRecord, extendedDataRecord.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
freezeFrame	DiagnosticFreezeFrame	*	ref	Define the links to a freeze frame class sampler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=freezeFrame.diagnosticFreezeFrame, freezeFrame.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
immediateNv DataStorage	Boolean	0..1	attr	Switch to enable immediate storage triggering of an according event memory entry persistently to NVRAM. true: immediate non-volatile storage triggering enabled false: immediate non-volatile storage triggering disabled
legislated FreezeFrame ContentUdsObd	DiagnosticDataIdentifier Set	0..1	ref	This reference identifies the layout of legislated freeze frames used for emission related diagnostics over the UDS protocol such as OBDOnUDS or WWH-OBD. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
maxNumber FreezeFrame Records	PositiveInteger	0..1	attr	This attribute defines the number of according freeze frame records, which can maximal be stored for this event. Therefore all these freeze frame records have the same freeze frame class.





Class	DiagnosticTroubleCodeProps			
priority	PositiveInteger	0..1	attr	Priority of the event, in view of full event buffer. A lower value means higher priority. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
significance	DiagnosticSignificanceEnum	0..1	attr	Significance of the event, which indicates additional information concerning fault classification and resolution.
snapshot RecordContent	DiagnosticDataIdentifierSet	0..1	ref	This represents the freeze frame layout as a set of DIDs. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.97: DiagnosticTroubleCodeProps

Class	DiagnosticTroubleCodeUds			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode			
Note	This element is used to describe non OBD-relevant DTCs. Tags: atp.recommendedPackage=DiagnosticTroubleCodes			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticTroubleCode, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
considerPtoStatus	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. True: the event is affected by the Dem PTO handling. False: the event is not affected by the Dem PTO handling.
dtcProps	DiagnosticTroubleCodeProps	0..1	ref	Defined properties associated with the DemDTC.
eventObdReadinessGroup	NameToken	0..1	attr	This attribute specifies the Event OBD Readiness group for PID \$01 and PID \$41 computation. This attribute is only applicable for emission-related ECUs.
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.
severity	DiagnosticUdsSeverityEnum	0..1	attr	DTC severity according to ISO 14229-1.
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
wwhObdDtcClass	DiagnosticWwhObdDtcClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OB DTC. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table A.98: DiagnosticTroubleCodeUds

Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition





Class	DiagnosticTroubleCodeUdsToClearConditionGroupMapping			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticMappings			
Base	ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticMapping , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Type	Mult.	Kind	Note
clearConditionGroup	DiagnosticClearConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClearConditionGroup. Tags: atp.Status=draft
troubleCodeUds	DiagnosticTroubleCodeUds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping. Tags: atp.Status=draft

Table A.99: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

Enumeration	DiagnosticTypeOfFreezeFrameRecordNumerationEnum
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	FreezeFrame record numeration type
Literal	Description
calculated	Freeze frame records will be numbered consecutive starting by 1 in their chronological order. Tags: atp.EnumerationLiteralIndex=0
configured	Freeze frame records will be numbered based on the given configuration in their chronological order. Tags: atp.EnumerationLiteralIndex=1

Table A.100: DiagnosticTypeOfFreezeFrameRecordNumerationEnum

Class	DiagnosticUploadInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for uploading data using diagnostic channels on the adaptive platform. Tags: atp.Status=draft atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.101: DiagnosticUploadInterface

Class	DiagnosticWriteDataByIdentifier			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::DataByIdentifier			
Note	This represents an instance of the "Write Data by Identifier" diagnostic service. Tags: atp.recommendedPackage=DiagnosticDataByIdentifiers			





Class	DiagnosticWriteDataByIdentifier			
Base	<i>ARElement, ARObject, CollectableElement, DiagnosticCommonElement, DiagnosticDataByIdentifier, DiagnosticServiceInstance, Identifiable, MultilanguageReferrable, PackageableElement, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
writeClass	DiagnosticWriteDataByIdentifierClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticWriteDataByIdentifier in the given context.

Table A.102: DiagnosticWriteDataByIdentifier

Class	DolpNetworkConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This element collects DoIP properties that are network interface specific. Tags: atp.Status=draft			
Base	<i>ARObject</i>			
Attribute	Type	Mult.	Kind	Note
eidUseMac	Boolean	0..1	attr	This attribute defines whether the MAC of the network interface is used as eid. True: MAC is used False: eid needs to be configured manually by DolpInstantiation.eid. Tags: atp.Status=draft
isActivationLineDependent	Boolean	1	attr	This attribute defines whether the network interface <ul style="list-style-type: none"> is started "on-demand" when an activation line is sensed or is always available. Tags: atp.Status=draft
maxInitialVehicleAnnouncementTime	TimeValue	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 13400-2:2012. The value of this timing shall be determined randomly in the closed interval [0..maxInitialVehicleAnnouncementTime]. Tags: atp.Status=draft
maxTesterConnections	PositiveInteger	1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed. Tags: atp.Status=draft
networkConfiguration	PlatformModuleEthernetEndpointConfiguration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN. Tags: atp.Status=draft
networkInterfaceId	PositiveInteger	1	attr	This attribute defines the identifier for the DoIPInterface. Tags: atp.Status=draft
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 13400-2:2012. Tags: atp.Status=draft





Class	DolpNetworkConfiguration			
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 13400-2:2012. Tags: atp.Status=draft
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 13400-2:2012. Tags: atp.Status=draft
vehicleAnnouncementCount	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2:2012. Tags: atp.Status=draft
vehicleAnnouncementInterval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2012. Tags: atp.Status=draft
vehicleIdentificationSyncStatus	Boolean	1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/announcement. Tags: atp.Status=draft

Table A.103: DolpNetworkConfiguration

Class	Identifiable (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	ARObject, MultilanguageReferrable, Referrable
Subclasses	ARPackage, AbstractDolpLogicAddressProps, AbstractEvent, AbstractImplementationDataTypeElement, AbstractSecurityEventFilter, AbstractSecurityIdsmInstanceFilter, AbstractServiceInstance, AbstractSignalBasedToSignalTriggeringMapping, AdaptiveModuleInstantiation, AdaptiveSwcInternalBehavior, ApApplicationEndpoint, ApplicationEndpoint, ApplicationError, ArtifactChecksum, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpFeature, AutosarOperationArgumentInstance, AutosarVariableInstance, BuildActionEntity, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, CollectableElement, ComManagementMapping, CommConnectorPort, CommunicationConnector, CommunicationController, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, CouplingPortStructuralElement, CryptoCertificate, CryptoKeySlot, CryptoProvider, CryptoServiceMapping, DataPrototypeGroup, DataTransformation, DdsDomainRange, DependencyOnArtifact, DeterministicClientResourceNeeds, DiagEventDebounceAlgorithm, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibitSource, DiagnosticRoutineSubfunction, DltApplication, DltArgument, DltMessage, DolpInterface, DolpLogicAddress, DolpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, EthernetWakeupSleepOnDatelineConfig, EventHandler, EventMapping, ExclusiveArea, ExecutableEntity, ExecutionTime, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMapping, FlexrayArTpNode, FlexrayTpPduPool, FrameTriggering, GeneralParameter, GlobalSupervision, GlobalTimeGateway, GlobalTimeMaster, GlobalTimeSlave, HealthChannel, HeapUsage, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPsecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, IdentCaption, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacMulticastGroup, McDataInstance, MemorySection, MethodMapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, NmCluster, NmNode, PackageableElement, ParameterAccess, PduActivationRouting





Class	Identifiable (abstract)			
	Group, PduToFrameMapping, PduTriggering, PerInstanceMemory, <i>PersistencyDeploymentElement</i> , <i>PersistencyInterfaceElement</i> , <i>PhmSupervision</i> , <i>PhysicalChannel</i> , PortGroup, <i>PortInterfaceMapping</i> , PossibleErrorReaction, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlot Mapping, RecoveryNotification, ResourceConsumption, ResourceGroup, RootSwClusterDesign ComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, Rpt Container, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptService Point, RunnableEntityGroup, <i>SdgAttribute</i> , SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, <i>SecureCommunicationDeployment</i> , SecureCommunication FreshnessProps, SecurityEventContextProps, <i>ServiceEventDeployment</i> , <i>ServiceFieldDeployment</i> , <i>ServiceInterfaceElementSecureComConfig</i> , <i>ServiceMethodDeployment</i> , <i>ServiceNeeds</i> , SignalService TranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, Someip EventGroup, SomeipProvidedEventGroup, SomeipTpChannel, <i>SpecElementReference</i> , <i>StackUsage</i> , StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionMode Condition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, SystemMemoryUsage, <i>TimeBaseResource</i> , TimingCondition, <i>TimingConstraint</i> , <i>TimingDescription</i> , TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, Tls JobMapping, Topic1, TpAddress, TraceableTable, TraceableText, <i>TracedFailure</i> , <i>TransformationProps</i> , TransformationTechnology, Trigger, UcmDescription, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
Attribute	Type	Mult.	Kind	Note
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. Stereotypes: atpSplitable Tags: atp.Splitkey=adminData xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. Tags: xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. Tags: xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". Tags: xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. Tags: xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open





Class	Identifiable (abstract)			
				<p>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.</p> <p>Tags:xml.attribute=true</p>

Table A.104: Identifiable

Class	ImplementationProps (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Implementation			
Note	Defines a symbol to be used as (depending on the concrete case) either a complete replacement or a prefix when generating code artifacts.			
Base	ARObject, Referrable			
Subclasses	BswSchedulerNamePrefix, ExecutableEntityActivationReason, SectionNamePrefix, SymbolProps , SymbolicNameProps			
Attribute	Type	Mult.	Kind	Note
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.105: ImplementationProps

Class	PPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	ARObject, AbstractProvidedPortPrototype, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, PortPrototype , Referrable			
Attribute	Type	Mult.	Kind	Note
provided Interface	PortInterface	0..1	tref	The interface that this port provides. Stereotypes: isOfType

Table A.106: PPortPrototype

Class	PortInterface (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable , MultilanguageReferrable, PackageableElement, Referrable			
Subclasses	AbstractRawDataStreamInterface, AbstractSynchronizedTimeBaseInterface, ClientServerInterface, CryptoInterface, DataInterface, DiagnosticPortInterface , LogAndTraceInterface, ModeSwitchInterface, PersistencyInterface, PlatformHealthManagementInterface, SecurityEventReportInterface, Service Interface, TriggerInterface			
Attribute	Type	Mult.	Kind	Note





Class	PortInterface (abstract)			
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. Stereotypes: atpSplittable Tags: atp.Splitkey=namespace.shortName atp.Status=draft

Table A.107: PortInterface

Class	PortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Subclasses	AbstractProvidedPortPrototype, AbstractRequiredPortPrototype			
Attribute	Type	Mult.	Kind	Note
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
logAndTrace Message CollectionSet	LogAndTraceMessage CollectionSet	0..1	ref	Reference to a collection of Log or Trace messages that will be used by the application. Tags: atp.Status=draft
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. Tags: atp.Status=draft
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.108: PortPrototype

Class	RPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	ARObject, AbstractRequiredPortPrototype, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, PortPrototype, Referrable			
Attribute	Type	Mult.	Kind	Note





Class	RPortPrototype			
required Interface	PortInterface	0..1	tref	The interface that this port requires. Stereotypes: isOfType

Table A.109: RPortPrototype

Class	SoftwareCluster			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. Tags: atp.Status=draft atp.recommendedPackage=SoftwareClusters			
Base	<i>ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, Packageable Element, Referrable</i>			
Attribute	Type	Mult.	Kind	Note
artifact Checksum	ArtifactChecksum	*	aggr	This aggregation carries the checksums for artifacts contained in the enclosing SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=artifactChecksum.shortName, artifact Checksum.uri atp.Status=draft
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 Tags: atp.Status=draft
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. Stereotypes: atpSplitable Tags: atp.Splitkey=conflictsTo atp.Status=draft
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=containedARElement atp.Status=draft
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster. Tags: atp.Status=draft
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. Stereotypes: atpSplitable Tags: atp.Splitkey=containedPackageElement atp.Status=draft
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. Tags: atp.Status=draft





Class	SoftwareCluster			
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=dependsOn atp.Status=draft
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all SoftwareClusterDesigns applicable for the enclosing SoftwareCluster. Stereotypes: atpUriDef Tags: atp.Status=draft
diagnosticExtract	DiagnosticContributionSet	0..1	ref	This reference represents the definition of the diagnostic extract applicable to the referencing SoftwareCluster Tags: atp.Status=draft
diagnosticProps	SoftwareClusterDiagnosticProps	0..1	aggr	This aggregation representst the diagnostic-related configuration of a SoftwareCluster. Tags: atp.Status=draft
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. Tags: atp.Status=draft
moduleInstantiation	AdaptiveModuleInstantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=moduleInstantiation atp.Status=draft
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. Tags: atp.Status=draft
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country. Tags: atp.Status=draft
vendorId	PositiveInteger	1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list. Tags: atp.Status=draft
vendorSignature	CryptoServiceCertificate	1	ref	This reference identifies the certificate that represents the vendor's signature. Tags: atp.Status=draft
version	StrongRevisionLabelString	1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster. Tags: atp.Status=draft

Table A.110: SoftwareCluster

Class	SoftwareClusterDiagnosticAddress (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address in an abstract form. Sub-classes are supposed to clarify how the diagnostic address shall be defined according to the applicable addressing scheme (DoIP vs. CAN TP vs. ...). Tags: atp.Status=draft			
Base	ARObject			
Subclasses	SoftwareClusterDoipDiagnosticAddress			
Attribute	Type	Mult.	Kind	Note
address Semantics	SoftwareClusterDiagnosticAddressSemanticsEnum	1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address. Tags: atp.Status=draft

Table A.111: SoftwareClusterDiagnosticAddress

Enumeration	SoftwareClusterDiagnosticAddressSemanticsEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster. Tags: atp.Status=draft
Literal	Description
functionalAddress	This address represents a functional address. Tags: atp.EnumerationLiteralIndex=1 atp.Status=draft
physicalAddress	This address represents a physical address. Tags: atp.EnumerationLiteralIndex=0 atp.Status=draft

Table A.112: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	SoftwareClusterDiagnosticProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class acts as the owner of all diagnostic-related properties of a SoftwareCluster. Tags: atp.Status=draft			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
diagnostic Address	SoftwareClusterDiagnosticAddress	*	aggr	This aggregation represents the collection of diagnostic addresses that apply for the SoftwareCluster. Stereotypes: atpSplitable Tags: atp.Splitkey=diagnosticAddress atp.Status=draft
external Authentication	DiagnosticExternalAuthenticationIdentification	*	aggr	This reference supports the configuration of the authentication of diagnostic clients. Tags: atp.Status=draft





Class		SoftwareClusterDiagnosticProps		
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. Tags: atp.Status=draft
validation Configuration	DiagnosticService ValidationConfiguration	0..1	aggr	This aggregation represents the ability to define the order of manufacturer and supplier validations in diagnostic management. Tags: atp.Status=draft

Table A.113: SoftwareClusterDiagnosticProps

Class		<<atpVariation>> SwDataDefProps		
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	<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"> • 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 Data Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTargetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout <p>Tags:vh.latestBindingTime=codeGenerationTime</p>			
Base	ARObject			
Attribute	Type	Mult.	Kind	Note
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. Tags: xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. Tags: xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. Tags: xml.sequenceOffset=50





Class	<<atpVariation>> SwDataDefProps			
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. Tags: xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. Tags: xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. Tags: xml.sequenceOffset=210
displayPresentation	DisplayPresentationEnum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementationDataType	AbstractImplementationDataType	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"> • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly Tags: xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. Tags: xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. Tags: xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. Tags: xml.sequenceOffset=33
swBitRepresentation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. Tags: xml.sequenceOffset=60
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	Specifies the read or write access by MCD tools for this data object. Tags: xml.sequenceOffset=70





Class	<<atpVariation>> SwDataDefProps			
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. Tags: xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. Tags: xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). Tags: xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. Tags: xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. Tags: xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". Tags: xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. Tags: xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . Tags: xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. Tags: xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. Tags: xml.sequenceOffset=290





Class	<<atpVariation>> SwDataDefProps			
swRefresh Timing	MultidimensionalTime	0..1	aggr	<p>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.</p> <p>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.</p> <p>Tags:xml.sequenceOffset=300</p>
swTextProps	SwTextProps	0..1	aggr	<p>the specific properties if the data object is a text object.</p> <p>Tags:xml.sequenceOffset=120</p>
swValueBlock Size	Numerical	0..1	attr	<p>This represents the size of a Value Block</p> <p>Stereotypes: atpVariation</p> <p>Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=80</p>
swValueBlock SizeMult (ordered)	Numerical	*	attr	<p>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.</p> <p>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.</p> <p>For one-dimensional value blocks the attribute swValueBlockSize shall be used and this attribute shall not exist.</p> <p>Stereotypes: atpVariation</p> <p>Tags:vh.latestBindingTime=preCompileTime</p>
unit	Unit	0..1	ref	<p>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.</p> <p>Tags:xml.sequenceOffset=350</p>
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	<p>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.</p> <p>Tags:xml.sequenceOffset=355</p>

Table A.114: SwDataDefProps

Class	SymbolProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	ARObject, ImplementationProps , Referrable			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table A.115: SymbolProps

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::ByteVector

[SWS_DM_00338]{DRAFT} [

Kind:	type alias
Symbol:	ByteVector
Scope:	namespace ara::diag::uds_transport
Derived from:	ara::core::Span<uint8_t>
Syntax:	using ByteVector = ara::core::Span<uint8_t>;
Header file:	#include "ara/diag/uds_transport/protocol_types.h"
Description:	This is the type of ByteVector.

]([RS_Diag_04147](#))

B.1.1.2 uds_transport::ChannelID

[SWS_DM_00337]{DRAFT} [

Kind:	type alias
Symbol:	ChannelID
Scope:	namespace ara::diag::uds_transport
Derived from:	uint32_t
Syntax:	using ChannelID = uint32_t;
Header file:	#include "ara/diag/uds_transport/protocol_types.h"
Description:	

]([RS_Diag_04147](#))

B.1.1.3 uds_transport::Priority

[SWS_DM_00451]{DRAFT} [

Kind:	type alias
Symbol:	Priority
Scope:	namespace ara::diag::uds_transport
Derived from:	uint8_t
Syntax:	using Priority = uint8_t;
Header file:	#include "ara/diag/uds_transport/protocol_types.h"
Description:	

|(RS_Diag_04147)

B.1.1.4 uds_transport::ProtocolKind

[SWS_DM_00452]{DRAFT} [

Kind:	type alias
Symbol:	ProtocolKind
Scope:	namespace ara::diag::uds_transport
Derived from:	ara::core::String
Syntax:	using ProtocolKind = ara::core::String;
Header file:	#include "ara/diag/uds_transport/protocol_types.h"
Description:	

|(RS_Diag_04147, RS_Diag_04168)

B.1.1.5 uds_transport::UdsMessageConstPtr

[SWS_DM_00304]{DRAFT} [

Kind:	type alias
Symbol:	UdsMessageConstPtr
Scope:	namespace ara::diag::uds_transport
Derived from:	std::unique_ptr<const UdsMessage, std::function<void(const UdsMessage*)> >
Syntax:	using UdsMessageConstPtr = std::unique_ptr<const UdsMessage, std::function<void(const UdsMessage*)> >;
Header file:	#include "ara/diag/uds_transport/uds_message.h"
Description:	This is the unique_ptr for constant UdsMessages containing a custom deleter as provided by the generic/core DM part towards the UdsTransportLayer-Plugin.
Notes:	How the exact typedef for UdsMessageConstPtr looks like, is up to the DM product vendor. I.e. how f.i. the deleter signature looks like ... basically the minimal agreement is: UdsMessage ConstPtr shall behave like a std::unique_ptr<const UdsMessage>!

|(RS_Diag_04147)

B.1.1.6 uds_transport::UdsMessagePtr

[SWS_DM_00303]{DRAFT} [

Kind:	type alias
Symbol:	UdsMessagePtr
Scope:	namespace ara::diag::uds_transport
Derived from:	std::unique_ptr<UdsMessage, std::function<void(UdsMessage*)>>
Syntax:	using UdsMessagePtr = std::unique_ptr<UdsMessage, std::function<void(UdsMessage*)>>;
Header file:	#include "ara/diag/uds_transport/uds_message.h"
Description:	This is the unique_ptr for UdsMessages containing a custom deleter as provided by the generic/core DM part towards the UdsTransportLayer-Plugin.
Notes:	How the exact typedef for UdsMessagePtr looks like, is up to the DM product vendor. I.e. how f.i. the deleter signature looks like ... basically the minimal agreement is: UdsMessagePtr shall behave like a std::unique_ptr<UdsMessage>!

]([RS_Diag_04147](#))

B.1.1.7 uds_transport::UdsTransportProtocolHandlerID

[SWS_DM_00336]{DRAFT} [

Kind:	type alias
Symbol:	UdsTransportProtocolHandlerID
Scope:	namespace ara::diag::uds_transport
Derived from:	uint8_t
Syntax:	using UdsTransportProtocolHandlerID = uint8_t;
Header file:	#include "ara/diag/uds_transport/protocol_types.h"
Description:	UdsTransportProtocolHandler are flexible "plugins", which need an identification.

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.2 UdsMessage Class

[SWS_DM_00291]{DRAFT} [

Kind:	class
Symbol:	UdsMessage
Scope:	namespace ara::diag::uds_transport
Syntax:	class UdsMessage {...};
Header file:	#include "ara/diag/uds_transport/uds_message.h"





Description:	<p>class represents an UDS message exchanged between DM generic core (UdsTransport ProtocolMgr) and a specific implementation of UdsTransportProtocolHandler on diagnostic request reception path or diagnostic response transmission path.</p> <p>UdsMessage provides the storage for UDS requests/responses. Instances of UdsMessage (with optimized resource allocation) are only created by DM generic core. UdsTransport ProtocolHandler read/write on it.</p>
---------------------	---

]([RS_Diag_04147](#))

B.1.2.1 Types

B.1.2.1.1 uds_transport::UdsMessage::Address

[SWS_DM_00293]{DRAFT} [

Kind:	type alias
Symbol:	Address
Scope:	class ara::diag::uds_transport::UdsMessage
Derived from:	uint16_t
Syntax:	using Address = uint16_t;
Header file:	#include "ara/diag/uds_transport/uds_message.h"
Description:	type for UDS source and target addresses

]([RS_Diag_04147](#))

B.1.2.1.2 uds_transport::UdsMessage::MetaInfoMap

[SWS_DM_00294]{DRAFT} [

Kind:	type alias
Symbol:	MetaInfoMap
Scope:	class ara::diag::uds_transport::UdsMessage
Derived from:	ara::core::Map<ara::core::String, ara::core::String>
Syntax:	using MetaInfoMap = ara::core::Map<ara::core::String, ara::core::String>;
Header file:	#include "ara/diag/uds_transport/uds_message.h"
Description:	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} [

Kind:	enumeration				
Symbol:	TargetAddressType				
Scope:	class ara::diag::uds_transport::UdsMessage				
Underlying type:	std::uint8_t				
Syntax:	enum class TargetAddressType : std::uint8_t {...};				
Values:	<table border="1"> <tr> <td>kPhysical= 0</td> <td>–</td> </tr> <tr> <td>kFunctional= 1</td> <td>–</td> </tr> </table>	kPhysical= 0	–	kFunctional= 1	–
kPhysical= 0	–				
kFunctional= 1	–				
Header file:	#include "ara/diag/uds_transport/uds_message.h"				
Description:	type of target address in UdsMessage				

]([RS_Diag_04147](#))

B.1.2.2 Methods

B.1.2.2.1 uds_transport::UdsMessage::UdsMessage

[SWS_DM_09012]{DRAFT} [

Kind:	function
Symbol:	UdsMessage()
Scope:	class ara::diag::uds_transport::UdsMessage
Visibility:	protected
Syntax:	UdsMessage ();
Header file:	#include "ara/diag/uds_transport/uds_message.h"
Description:	non public default ctor. The default ctor is protected as we want to forbid, that UdsTransport Protocol handlers do create UdsMessages on its own! Only DM is allowed to create and hands over UdsMessagePtrs to UdsTransportProtocolHandler.

]([RS_Diag_04147](#))

B.1.2.2.2 uds_transport::UdsMessage::UdsMessage

[SWS_DM_09011]{DRAFT} [

Kind:	function		
Symbol:	UdsMessage(const UdsMessage &other)		
Scope:	class ara::diag::uds_transport::UdsMessage		
Visibility:	protected		
Syntax:	<code>UdsMessage (const UdsMessage &other)=default;</code>		
Parameters (in):	<table border="1"> <tr> <td>other</td> <td>Object to copy-construct from</td> </tr> </table>	other	Object to copy-construct from
other	Object to copy-construct from		
Thread Safety:	reentrant		
Header file:	#include "ara/diag/uds_transport/uds_message.h"		





Description:	Copy constructing the uds message.
---------------------	------------------------------------

](RS_Diag_04147)

B.1.2.2.3 uds_transport::UdsMessage::UdsMessage

[SWS_DM_09013]{DRAFT} [

Kind:	function	
Symbol:	UdsMessage(UdsMessage &&other)	
Scope:	class ara::diag::uds_transport::UdsMessage	
Visibility:	protected	
Syntax:	UdsMessage (UdsMessage &&other) noexcept=default;	
Parameters (in):	other	Object to move-construct from
Exception Safety:	noexcept	
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Move constructing the uds message.	

](RS_Diag_04147)

B.1.2.2.4 uds_transport::UdsMessage::UdsMessage::operator=

[SWS_DM_09014]{DRAFT} [

Kind:	function	
Symbol:	operator=(const UdsMessage &other)	
Scope:	class ara::diag::uds_transport::UdsMessage	
Visibility:	protected	
Syntax:	UdsMessage& operator= (const UdsMessage &other)=default;	
Parameters (in):	other	Object to copy-assign from.
Return value:	UdsMessage &	-
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Copy assigning the uds message.	

](RS_Diag_04147)

B.1.2.2.5 uds_transport::UdsMessage::UdsMessage::operator=

[SWS_DM_09018]{DRAFT} [

Kind:	function	
Symbol:	operator=(UdsMessage &&other)	
Scope:	class ara::diag::uds_transport::UdsMessage	
Visibility:	protected	
Syntax:	UdsMessage& operator= (UdsMessage &&other) noexcept=default;	
Parameters (in):	other	Object to move-assign from.
Return value:	UdsMessage &	–
Exception Safety:	noexcept	
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Move assigning the uds message.	

](RS_Diag_04147)

B.1.2.2.6 uds_transport::UdsMessage::~~UdsMessage

[SWS_DM_09010]{DRAFT} [

Kind:	function	
Symbol:	~UdsMessage()	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual ~UdsMessage ();	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Destructing the uds message.	

](RS_Diag_04147)

B.1.2.2.7 uds_transport::UdsMessage::AddMetaInfo

[SWS_DM_00302]{DRAFT} [

Kind:	function	
Symbol:	AddMetaInfo(MetaInfoMap metaInfo)	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual void AddMetaInfo (MetaInfoMap metaInfo);	
Parameters (in):	metaInfo	meta information relevant for UdsMessage
Return value:	None	
Thread Safety:	unsafe	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	add new metaInfo to this message.	
Notes:	typically called by the transport plugin to add channel specific meta-info. (see SWS - there are already predefined meta-info keys for DoIP...)	

|(RS_Diag_04147, RS_Diag_04170)

B.1.2.2.8 uds_transport::UdsMessage::GetPayload

[SWS_DM_00300]{DRAFT} [

Kind:	function	
Symbol:	GetPayload()	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual const uds_transport::ByteVector& GetPayload () const;	
Return value:	const uds_transport::ByteVector &	The entire payload (A_Data)
Thread Safety:	unsafe	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Get the UDS message data starting with the SID (A_Data as per ISO)	
Notes:	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} [

Kind:	function	
Symbol:	GetPayload()	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual uds_transport::ByteVector& GetPayload ();	
Return value:	uds_transport::ByteVector &	payload of the UDSMessage starting from SID.
Thread Safety:	unsafe	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	return the underlying buffer for write access.	
Notes:	<p>needed by UdsTransportProtocolHandler impl. to fill the UdsMessage with data in RX path. I.e. UdsTransportProtocolHandler impl. gets the UdsMessage instance from call to UdsTransportProtocolMgr::IndicateMessage() and then calls this method on it and write into returned uds_transport::ByteVector.</p> <p>marked as "unsafe" with regard to threadsafety as implementation is allowed to do ressource allocation of buffer in the context of this call.</p>	

|(RS_Diag_04147)

B.1.2.2.9 uds_transport::UdsMessage::GetSa

[SWS_DM_00297]{DRAFT} [

Kind:	function	
Symbol:	GetSa()	





Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual Address GetSa () const noexcept;	
Return value:	Address	The source address of the uds message.
Exception Safety:	noexcept	
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Get the source address of the uds message.	

]([RS_Diag_04147](#), [RS_Diag_04174](#))

B.1.2.2.10 uds_transport::UdsMessage::GetTa

[SWS_DM_00298]{DRAFT} [

Kind:	function	
Symbol:	GetTa()	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual Address GetTa () const noexcept;	
Return value:	Address	The target address of the uds message.
Exception Safety:	noexcept	
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	Get the target address of the uds message.	

]([RS_Diag_04147](#), [RS_Diag_04174](#))

B.1.2.2.11 uds_transport::UdsMessage::GetTaType

[SWS_DM_00299]{DRAFT} [

Kind:	function	
Symbol:	GetTaType()	
Scope:	class ara::diag::uds_transport::UdsMessage	
Syntax:	virtual TargetAddressType GetTaType () const noexcept;	
Return value:	TargetAddressType	The target address type of the uds message.
Exception Safety:	noexcept	
Thread Safety:	reentrant	
Header file:	#include "ara/diag/uds_transport/uds_message.h"	
Description:	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} [

Kind:	class
Symbol:	UdsTransportProtocolHandler
Scope:	namespace ara::diag::uds_transport
Syntax:	<code>class UdsTransportProtocolHandler {...};</code>
Header file:	<code>#include "ara/diag/uds_transport/protocol_handler.h"</code>
Description:	Abstract Class, which a specific UDS Transport Protocol (plugin) shall subclass.

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.3.1 Types

B.1.3.1.1 uds_transport::UdsTransportProtocolHandler::InitializationResult

[SWS_DM_09017]{DRAFT} [

Kind:	enumeration				
Symbol:	InitializationResult				
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler				
Underlying type:	std::uint8_t				
Syntax:	<code>enum class InitializationResult : std::uint8_t {...};</code>				
Values:	<table border="1"> <tr> <td>kInitializeOk= 0</td> <td>-</td> </tr> <tr> <td>kInitializeFailed= 1</td> <td>-</td> </tr> </table>	kInitializeOk= 0	-	kInitializeFailed= 1	-
kInitializeOk= 0	-				
kInitializeFailed= 1	-				
Header file:	<code>#include "ara/diag/uds_transport/protocol_handler.h"</code>				
Description:	Result of Initialize handler.				

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.3.2 Methods

B.1.3.2.1 uds_transport::UdsTransportProtocolHandler::UdsTransportProtocolHandler

[SWS_DM_09015]{DRAFT} [

Kind:	function
Symbol:	<code>UdsTransportProtocolHandler(const UdsTransportProtocolHandlerID handlerId, UdsTransportProtocolMgr &transportProtocolMgr)</code>
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler





Syntax:	explicit UdsTransportProtocolHandler (const UdsTransportProtocolHandlerID handlerId, UdsTransportProtocolMgr &transportProtocolMgr);	
Parameters (in):	handlerId	the handler ID used by DM to identify this handler. This is just a number/identification given by the DM core when instantiating a UdsTransportProtocolHandler instance to be able to distinguish it from other handler-plugins or built-in UdsTransportProtocolHandler implementations.
	transportProtocolMgr	reference to UdsTransportProtocolMgr owned by this DM, with which UdsTransportProtocolHandler instance shall interact.
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	Constructor of UdsTransportProtocolHandler.	

](RS_Diag_04147, RS_Diag_04168)

B.1.3.2.2 uds_transport::UdsTransportProtocolHandler::~~UdsTransport

[SWS_DM_09016]{DRAFT} [

Kind:	function
Symbol:	~UdsTransportProtocolHandler()
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler
Syntax:	virtual ~UdsTransportProtocolHandler ();
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"
Description:	Destructor of UdsTransportProtocolHandler.

](RS_Diag_04147, RS_Diag_04168)

B.1.3.2.3 uds_transport::UdsTransportProtocolHandler::GetHandlerID

[SWS_DM_00325]{DRAFT} [

Kind:	function	
Symbol:	GetHandlerID()	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler	
Syntax:	virtual UdsTransportProtocolHandlerID GetHandlerID () const;	
Return value:	UdsTransportProtocolHandlerID	UdsTransportProtocolHandlerID.
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	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} [

Kind:	function	
Symbol:	Initialize()	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler	
Syntax:	virtual InitializationResult Initialize ()=0;	
Return value:	InitializationResult	kInitializeOk if initialization was successful, else kInitializeFailed.
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	Initializes handler. Must be called before Start(). The idea is to have "initialization" of handler-plugin separated from its ctor.	

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.3.2.5 uds_transport::UdsTransportProtocolHandler::NotifyReestablishment

[SWS_DM_00326]{DRAFT} [

Kind:	function	
Symbol:	NotifyReestablishment(ChannelID channelId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler	
Syntax:	virtual bool NotifyReestablishment (ChannelID channelId)=0;	
Parameters (in):	channelId	channelID, whose re-establishment shall be notified to UdsTransportProtocolMgr
Return value:	bool	true if notification request is accepted and can be fulfilled.
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	Tells the UdsTransportProtocolHandler, that it shall notify the DM core via UdsTransportProtocolMgr::ChannelReestablished() if the given channel has been re-established after next UdsTransportProtocolHandler::Start(). The main purpose of this method is to allow DM to provide an ECU-Reset (0x11 service), with configuration option "Pos. 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 UdsProtocolHandlers/plugins get restarted via call to UdsTransportProtocolHandler::Start(). Now there are two expectations, when this method has been called before and returned "true": IF the same remote client connects to the UdsProtocolHandler, it shall get a channel identification with the same identifying tuple <p_x, c_y> as last time. it shall call UdsTransportProtocolMgr::ChannelReestablished(GlobalChannelIdentifier<p_x, c_y>)	
Notes:	: IF the underlying network layer of the UdsTransportProtocolHandler isn't really connection based (e.g. a UDP based protocol), then the UdsTransportProtocolHandler shall call UdsTransportProtocolMgr::ChannelReestablished() after UdsTransportProtocolHandler::Start() as soon as it detects/assumes that the remote client/tester will be reachable again. : The detection/decision, whether the "same" client reconnects as before is an UdsProtocolHandler implementation specific decision. The general expectation is: If the channel is set up from exactly the same remote network-endpoint, it typically shall be given the same channelId	





	<p style="text-align: center;">△</p> <p>(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 channelID (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 NotifyReestablishment() 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 only shall be 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.</p>
--	---

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.3.2.6 uds_transport::UdsTransportProtocolHandler::Start

[SWS_DM_00322]{DRAFT} [

Kind:	function
Symbol:	Start()
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler
Syntax:	virtual void Start ()=0;
Return value:	None
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"
Description:	<p>Start processing the implemented Uds Transport Protocol.</p> <p>The implementation shall call its superclass Start() method as there might be some stack specific implementation. Implementation shall be asynchronous as DM might start many/ different UdsTransportProtocolHandler in parallel and strong serialization of all those starts just unnecessarily slows down DM startup.</p>

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.3.2.7 uds_transport::UdsTransportProtocolHandler::Stop

[SWS_DM_00323]{DRAFT} [

Kind:	function
Symbol:	Stop()
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler
Syntax:	virtual void Stop ()=0;
Return value:	None
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"





Description:	<p>Method to indicate that this UdsTransportProtocolHandler should terminate.</p> <p>If UdsTransportProtocolHandler has stopped, it shall call UdsTransportProtocolMgr::HandlerStopped(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} [

Kind:	function	
Symbol:	Transmit(UdsMessageConstPtr message, ChannelID channelId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler	
Syntax:	virtual void Transmit (UdsMessageConstPtr message, ChannelID channelId)=0;	
Parameters (in):	message	The message to be transmitted as a Uds Message::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.
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	Transmit a Uds message via the underlying Uds Transport Protocol channel. This transmit API covers T_Data.req of ISO 14229-2 Figure 2.	

](RS_Diag_04147, RS_Diag_04168)

B.1.3.2.9 uds_transport::UdsTransportProtocolHandler::GetPeriodicHandler

[SWS_DM_01068]{DRAFT} [

Kind:	function	
Symbol:	GetPeriodicHandler()	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolHandler	
Syntax:	virtual ara::core::Result<UdsTransportProtocolPeriodicHandler&> GetPeriodicHandler ();	
Return value:	ara::core::Result< UdsTransportProtocolPeriodicHandler & >	UdsTransportProtocolPeriodicHandler reference if periodic transmissions are supported on this transport protocol, an error if not supported
Header file:	#include "ara/diag/uds_transport/protocol_handler.h"	
Description:	Returns the corresponding periodic TP handler.	

]([RS_Diag_04215](#))

B.1.4 UdsTransportProtocolMgr Class

[SWS_DM_00306]{DRAFT} [

Kind:	class
Symbol:	UdsTransportProtocolMgr
Scope:	namespace ara::diag::uds_transport
Syntax:	class UdsTransportProtocolMgr {...};
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"
Description:	

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.4.1 Types

B.1.4.1.1 uds_transport::UdsTransportProtocolMgr::GlobalChannelIdentifier

[SWS_DM_09021]{DRAFT} [

Kind:	type alias
Symbol:	GlobalChannelIdentifier
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr
Derived from:	std::tuple<UdsTransportProtocolHandlerID, ChannelID>
Syntax:	using GlobalChannelIdentifier = std::tuple<UdsTransportProtocolHandlerID, ChannelID>;
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"
Description:	Type of tuple to pack UdsTransportProtocolHandlerID and ChannelID together, to form a global unique (among all used UdsTransportProtocolHandlers within DM) identifier of a UdsTransport Protocol channel.

]([RS_Diag_04147](#), [RS_Diag_04168](#))

B.1.4.1.2 uds_transport::UdsTransportProtocolMgr::IndicationResult

[SWS_DM_00384]{DRAFT} [

Kind:	enumeration
Symbol:	IndicationResult
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr
Underlying type:	std::uint8_t





Syntax:	enum class IndicationResult : std::uint8_t {...};	
Values:	kIndicationOk= 0	–
	kIndicationOccupied= 1	–
	kIndicationOverflow= 2	–
	kIndicationUnknownTargetAddress= 3	–
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:		

|(RS_Diag_04147, RS_Diag_04168)

B.1.4.1.3 uds_transport::UdsTransportProtocolMgr::TransmissionResult

[SWS_DM_00307]{DRAFT} [

Kind:	enumeration	
Symbol:	TransmissionResult	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Underlying type:	std::uint8_t	
Syntax:	enum class TransmissionResult : std::uint8_t {...};	
Values:	kTransmitOk= 0	–
	kTransmitFailed= 1	–
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:		

|(RS_Diag_04172, RS_Diag_04147, RS_Diag_04168)

B.1.4.2 Methods

B.1.4.2.1 uds_transport::UdsTransportProtocolMgr::ChannelReestablished

[SWS_DM_00313]{DRAFT} [

Kind:	function	
Symbol:	ChannelReestablished(GlobalChannelIdentifier globalChannelId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void ChannelReestablished (GlobalChannelIdentifier globalChannelId)=0;	
Parameters (in):	globalChannelId	transport protocol channel, which is available again.
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	





Description:	notification call from the given transport channel, that it has been reestablished since the last (Re)Start from the UdsTransportProtocolHandler to which this channel belongs. To activate this notification a previous call to UdsTransportProtocolHandler::NotifyReestablishment() has to be done. See further documentation at UdsTransportProtocolHandler::NotifyReestablishment().
---------------------	--

|(RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_Diag_04147, RS_Diag_04168)

B.1.4.2.2 uds_transport::UdsTransportProtocolMgr::HandleMessage

[SWS_DM_00311]{DRAFT} [

Kind:	function	
Symbol:	HandleMessage(UdsMessagePtr message)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void HandleMessage (UdsMessagePtr message)=0;	
Parameters (in):	message	The Uds message ptr (unique_ptr semantics) with the request. Ownership of the UdsMessage is given back to the generic DM core here.
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:	Hands over a valid received Uds message (currently this is only a request type) from transport layer to session layer. It corresponds to T_Data.ind of Figure 2 from ISO 14229-2. The behavior is asynchronously. I.e. the UdsMessage is handed over to Session Layer and it is expected, that it "instantly" returns, which means, that real processing of the message shall be done asynchronously!	

|(RS_Diag_04147, RS_Diag_04168)

B.1.4.2.3 uds_transport::UdsTransportProtocolMgr::HandlerStopped

[SWS_DM_00314]{DRAFT} [

Kind:	function	
Symbol:	HandlerStopped(UdsTransportProtocolHandlerID handlerId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void HandlerStopped (UdsTransportProtocolHandlerID handlerId)=0;	
Parameters (in):	handlerId	indication, which plugin stopped.
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:	notification from handler, that it has stopped now (e.g. closed down network connections, freed resources, etc...) This callback is expected as a reaction from handler to a call to UdsTransportProtocolHandler::Stop.	

|(RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_Diag_04147, RS_Diag_04168)

B.1.4.2.4 uds_transport::UdsTransportProtocolMgr::IndicateMessage

[SWS_DM_00309]{DRAFT} [

Kind:	function	
Symbol:	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)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual std::pair<IndicationResult, UdsMessagePtr> IndicateMessage (UdsMessage::Address sourceAddr, UdsMessage::Address targetAddr, UdsMessage::TargetAddressType type, GlobalChannelIdentifier globalChannelId, std::size_t size, Priority priority, ProtocolKind protocolKind, ara::core::Span< const std::uint8_t > payloadInfo)=0;	
Parameters (in):	sourceAddr	UDS source address of message
	targetAddr	UDS target address of message
	type	indication whether its is phys/func request
	globalChannelId	transport protocol channel on which message start happened
	size	size in bytes of the UdsMessage starting from SID.
	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.
Return value:	std::pair< IndicationResult, Uds MessagePtr >	Pair of IndicationResult and a pointer to Uds Message owned/created by DM core and returned to the handler to get filled.
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:	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.1.4.2.5 uds_transport::UdsTransportProtocolMgr::NotifyMessageFailure

[SWS_DM_00310]{DRAFT} [

Kind:	function	
Symbol:	NotifyMessageFailure(UdsMessagePtr message)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void NotifyMessageFailure (UdsMessagePtr message)=0;	
Parameters (in):	message	the pointer to UdsMessage handed back over to the session layer.
Return value:	None	





Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"
Description:	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.1.4.2.6 uds_transport::UdsTransportProtocolMgr::TransmitConfirmation

[SWS_DM_00312]{DRAFT} [

Kind:	function	
Symbol:	TransmitConfirmation(UdsMessageConstPtr message, TransmissionResult result)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void TransmitConfirmation (UdsMessageConstPtr message, TransmissionResult result)=0;	
Parameters (in):	message	for which message (created in IndicateMessage()) this is the confirmation.
	result	Result of transmission. In case UDS message could be transmitted on network layer: kTransmitOk, kTransmitFailed else.
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"	
Description:	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.1.4.2.7 uds_transport::UdsTransportProtocolMgr::PeriodicTransmitConfirmation

[SWS_DM_01069]{DRAFT} [

Kind:	function	
Symbol:	PeriodicTransmitConfirmation(ara::core::Vector< UdsMessageConstPtr > messages, std::size_t numberOfSentMessages)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolMgr	
Syntax:	virtual void PeriodicTransmitConfirmation (ara::core::Vector< UdsMessageConstPtr > messages, std::size_t numberOfSentMessages)=0;	
Parameters (in):	messages	The same ordered list of messages previously passed to UdsTransportProtocolPeriodicHandler::PeriodicTransmit.
	numberOfSentMessages	The number of successfully sent messages from the "messages" list.
Return value:	None	



△

Header file:	#include "ara/diag/uds_transport/protocol_mgr.h"
Description:	Confirmation of sent messages and number.

](RS_Diag_04215)

B.1.5 UdsTransportProtocolPeriodicHandler Class

[SWS_DM_01064]{DRAFT} [

Kind:	class
Symbol:	UdsTransportProtocolPeriodicHandler
Scope:	namespace ara::diag::uds_transport
Syntax:	class UdsTransportProtocolPeriodicHandler {...};
Header file:	#include "ara/diag/uds_transport/protocol_periodic_handler.h"
Description:	UdsTransportProtocolPeriodicHandler class to support 0x2A service from ISO.

](RS_Diag_04215)

B.1.5.1 Methods

B.1.5.1.1 uds_transport::UdsTransportProtocolPeriodicHandler::GetNumberOfPeriodicMessages

[SWS_DM_01065]{DRAFT} [

Kind:	function	
Symbol:	GetNumberOfPeriodicMessages(ChannelID channelId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler	
Syntax:	virtual std::size_t GetNumberOfPeriodicMessages (ChannelID channelId) const =0;	
Parameters (in):	channelId	The concrete connection to send over
Return value:	std::size_t	number of periodic messages
Header file:	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
Description:	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} [

Kind:	function	
Symbol:	GetMaxPayloadLength(ChannelID channelId)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler	
Syntax:	virtual std::size_t GetMaxPayloadLength (ChannelID channelId) const =0;	
Parameters (in):	channelId	The concrete connection for which the maximum payload length is reported.
Return value:	std::size_t	supported payload length
Header file:	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
Description:	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} [

Kind:	function	
Symbol:	PeriodicTransmit(ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages)	
Scope:	class ara::diag::uds_transport::UdsTransportProtocolPeriodicHandler	
Syntax:	virtual void PeriodicTransmit (ChannelID channelId, ara::core::Vector< UdsMessageConstPtr > messages)=0;	
Parameters (in):	channelId	The concrete connection to send over
	messages	Ordered list of messages to send at once
Return value:	None	
Header file:	#include "ara/diag/uds_transport/protocol_periodic_handler.h"	
Description:	Sends all the messages in the list in the given order. If one message transmission fails, the send process is stopped and the PeriodicTransmitConfirmation is invoked with the number of sent messages and the same Vector with UdsMessages.	

](RS_Diag_04215)

B.1.6 Sequence Diagrams of UDS Transport Layer Interaction

B.1.6.1 Lifecycle

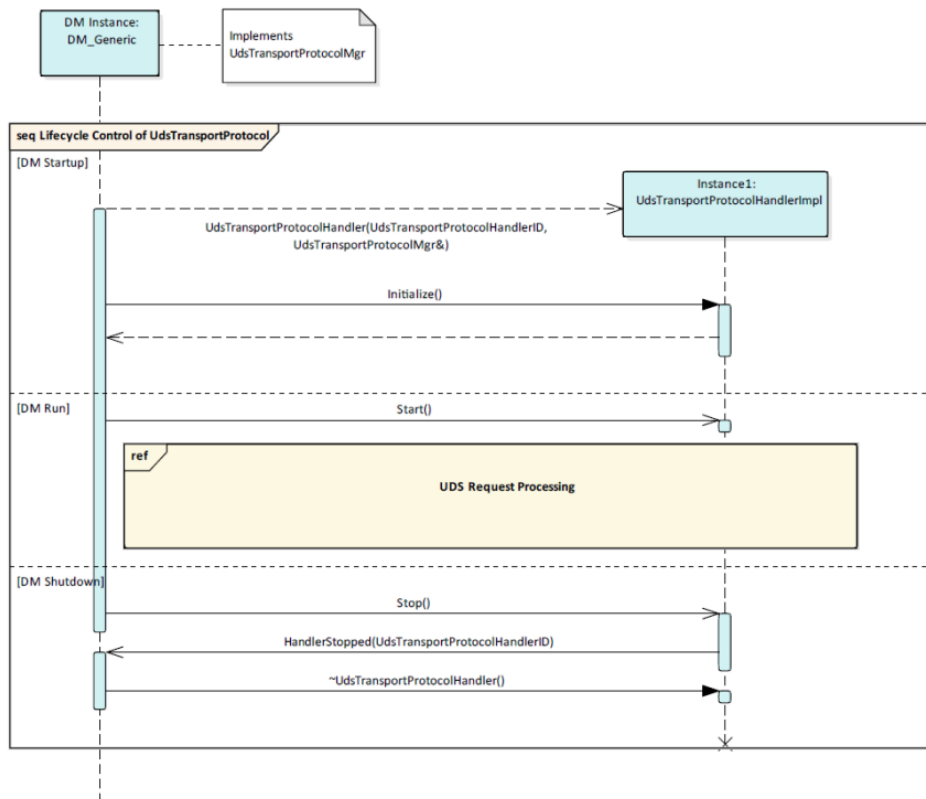


Figure B.1: UDS Transport Lifecycle

B.1.6.2 UDS Request Processing

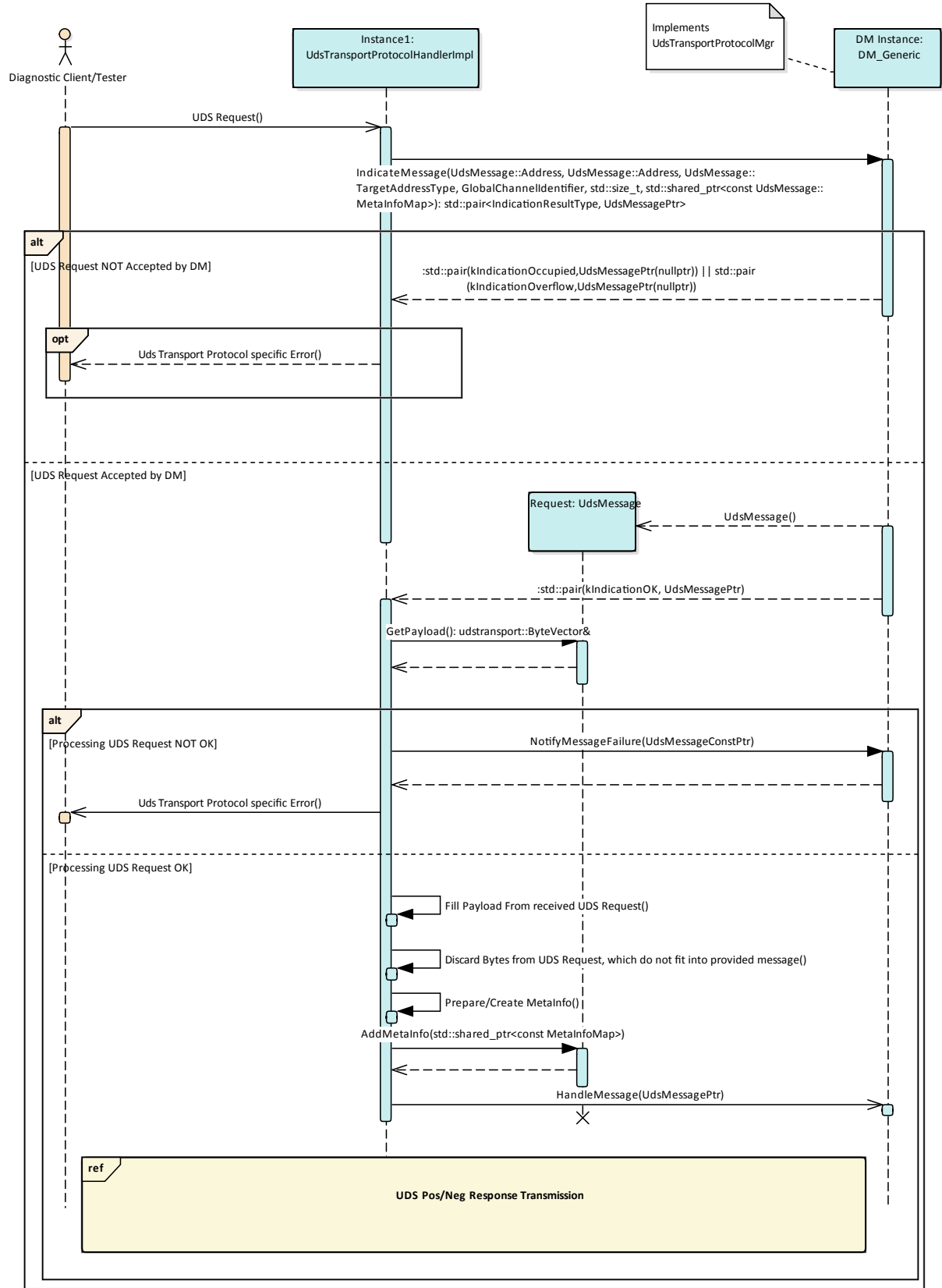
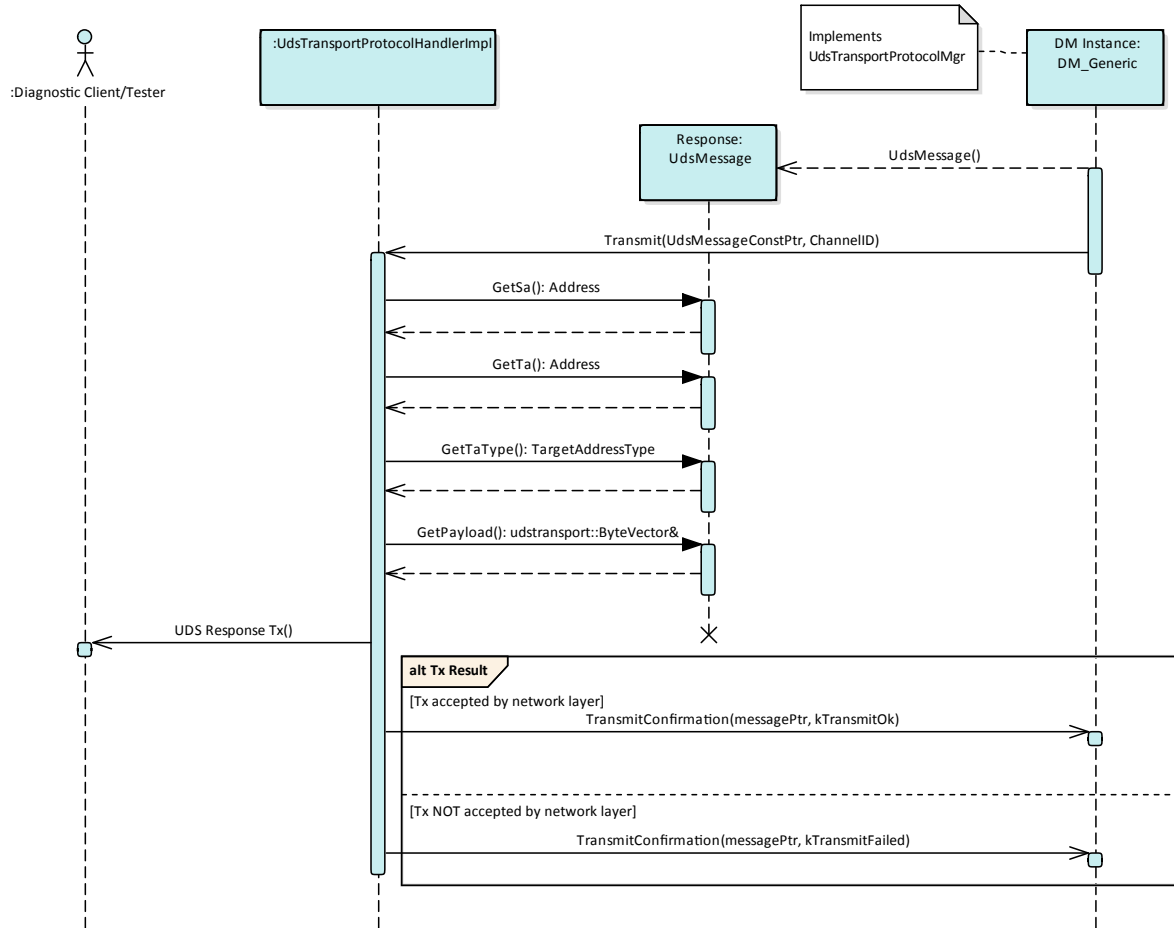
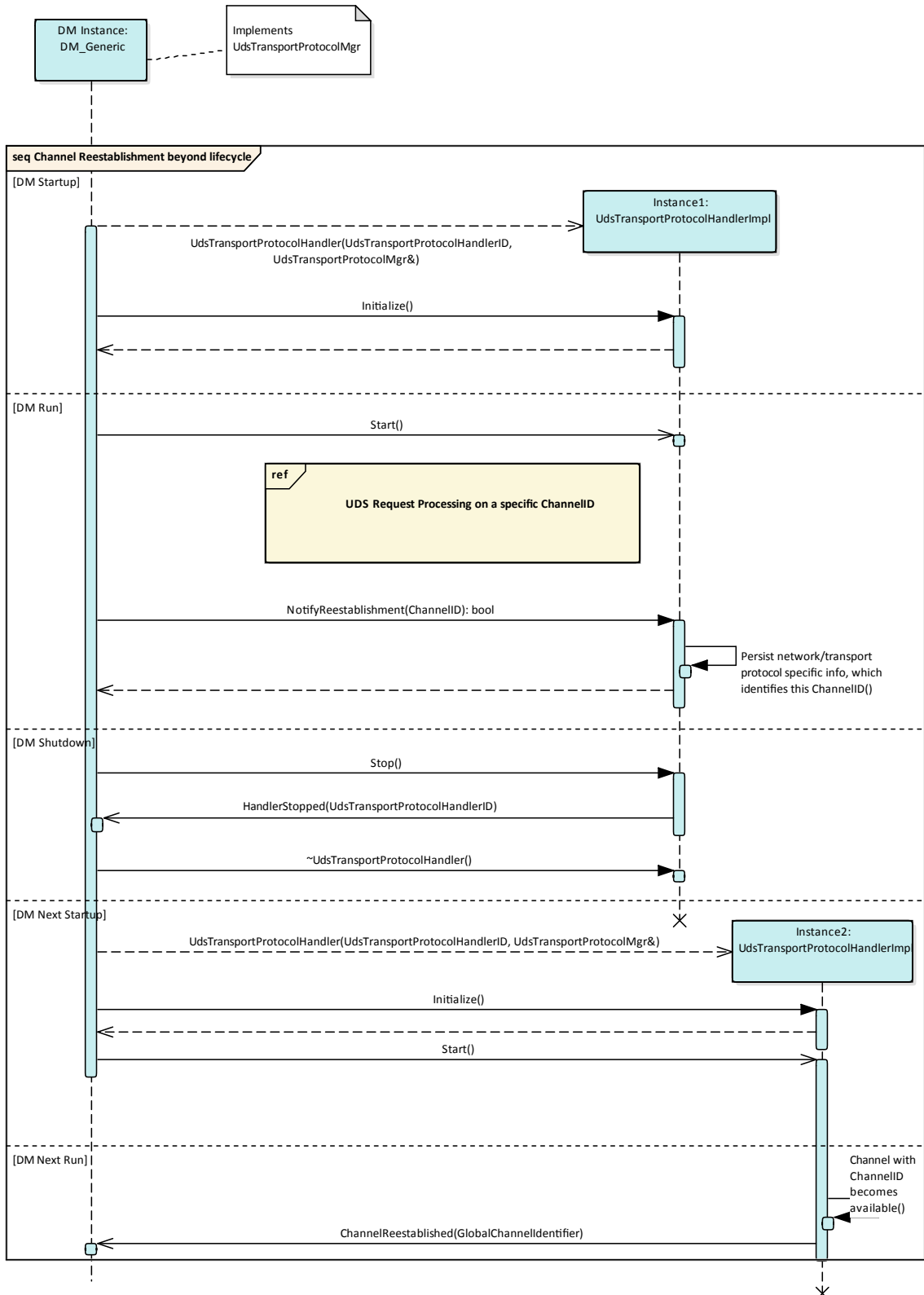


Figure B.2: UDS Transport Request Processing

B.1.6.3 UDS Response Transmission



B.1.6.4 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} [

Kind:	class
Symbol:	DoIPGroupIdentification
Scope:	namespace ara::diag
Syntax:	class DoIPGroupIdentification {...};
Header file:	#include "ara/diag/doip_group_identification.h"
Description:	DoIPGroupIdentificationInterface.

]([SRS_Eth_00026](#))

B.2.1.1 diag::DoIPGroupIdentification::DoIPGroupIdentificationType type

[SWS_DM_00721]{DRAFT} [

Kind:	struct
Symbol:	GidStatus
Scope:	class ara::diag::DoIPGroupIdentification
Syntax:	struct GidStatus {...};
Header file:	#include "ara/diag/doip_group_identification.h"
Description:	Response data of positive response message.

]([SRS_Eth_00026](#))

B.2.1.2 diag::DoIPGroupIdentification::DoIPGroupIdentification function

[SWS_DM_00722]{DRAFT} [

Kind:	function	
Symbol:	DoIPGroupIdentification(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::DoIPGroupIdentification	
Syntax:	explicit DoIPGroupIdentification (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPGroupIdentificationInterface





	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/doip_group_identification.h"	
Description:	Constructor of DoIPGroupIdentification.	

]([RS_AP_00137](#), [SRS_Eth_00026](#))

B.2.1.3 diag::DoIPGroupIdentification::~~DoIPGroupIdentification function

[SWS_DM_00723]{DRAFT} [

Kind:	function	
Symbol:	~DoIPGroupIdentification()	
Scope:	class ara::diag::DoIPGroupIdentification	
Syntax:	virtual ~DoIPGroupIdentification () noexcept=default;	
Exception Safety:	noexcept	
Header file:	#include "ara/diag/doip_group_identification.h"	
Description:	Destructor of DoIPGroupIdentification.	

]([RS_AP_00134](#), [SRS_Eth_00026](#))

B.2.1.4 diag::DoIPGroupIdentification::GetGidStatus function

[SWS_DM_00724]{DRAFT} [

Kind:	function	
Symbol:	GetGidStatus()	
Scope:	class ara::diag::DoIPGroupIdentification	
Syntax:	virtual ara::core::Future<GidStatus> GetGidStatus ()=0;	
Return value:	ara::core::Future< GidStatus >	group identification and state
Header file:	#include "ara/diag/doip_group_identification.h"	
Description:	Called to get the current GID state for the DoIP protocol.	

]([RS_AP_00138](#), [SRS_Eth_00026](#))

B.2.1.5 diag::DoIPGroupIdentification::Offer function

[SWS_DM_00725]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::DoIPGroupIdentification	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/doip_group_identification.h"	
Description:	This Offer will enable the DM to forward request messages to this handler.	

|(RS_AP_00119, RS_AP_00139, SRS_Eth_00026)

B.2.1.6 diag::DoIPGroupIdentification::StopOffer function

[SWS_DM_00726]{DRAFT} [

Kind:	function	
Symbol:	StopOffer()	
Scope:	class ara::diag::DoIPGroupIdentification	
Syntax:	void StopOffer ();	
Return value:	None	
Header file:	#include "ara/diag/doip_group_identification.h"	
Description:	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} [

Kind:	class	
Symbol:	DoIPPowerMode	
Scope:	namespace ara::diag	
Syntax:	class DoIPPowerMode {...};	
Header file:	#include "ara/diag/doip_power_mode.h"	
Description:	DiagnosticDoIPPowerModeInterface.	

|(SRS_Eth_00080)

B.2.2.1 diag::PowerModeType type

[SWS_DM_00730]{DRAFT} [

Kind:	enumeration	
Symbol:	PowerModeType	
Scope:	namespace ara::diag	
Underlying type:	–	
Syntax:	enum class PowerModeType {...};	
Values:	kNotReady= 0x00	not all ECUs accessible via DoIP can communicate
	kReady= 0x01	all ECUs accessible via DoIP can communicate
	kNotSupported= 0x02	the Diagnostic Information Power Mode Information Request message is not supported
Header file:	#include "ara/diag/doip_power_mode.h"	
Description:	PowerMode as defined in ISO13400-2.	

]([SRS_Eth_00080](#))

B.2.2.2 diag::DoIPPowerMode::DoIPPowerMode function

[SWS_DM_00732]{DRAFT} [

Kind:	function	
Symbol:	DoIPPowerMode(const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType)	
Scope:	class ara::diag::DoIPPowerMode	
Syntax:	explicit DoIPPowerMode (const ara::core::InstanceSpecifier &specifier, ReentrancyType reentrancyType);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPPowerModeInterface
	reentrancyType	specifies if interface is callable fully- or non-reentrant
Header file:	#include "ara/diag/doip_power_mode.h"	
Description:	Constructor of DoIPPowerMode.	

]([RS_AP_00137](#), [SRS_Eth_00080](#))

B.2.2.3 diag::DoIPPowerMode::~~DoIPPowerMode function

[SWS_DM_00733]{DRAFT} [

Kind:	function
Symbol:	~DoIPPowerMode()
Scope:	class ara::diag::DoIPPowerMode



△

Syntax:	<code>virtual ~DoIPPowerMode () noexcept=default;</code>
Exception Safety:	<code>noexcept</code>
Header file:	<code>#include "ara/diag/doip_power_mode.h"</code>
Description:	Destructor of DoIPPowerMode.

]([RS_AP_00134](#), [SRS_Eth_00080](#))

B.2.2.4 diag::DoIPPowerMode::GetDoIPPowerMode function

[SWS_DM_00734]{DRAFT} [

Kind:	function	
Symbol:	GetDoIPPowerMode()	
Scope:	class ara::diag::DoIPPowerMode	
Syntax:	<code>virtual ara::core::Future<PowerModeType> GetDoIPPowerMode ()=0;</code>	
Return value:	ara::core::Future< PowerModeType >	current diagnostic power mode
Header file:	<code>#include "ara/diag/doip_power_mode.h"</code>	
Description:	Called to get the current Power Mode for the DoIP protocol.	

]([RS_AP_00138](#), [SRS_Eth_00080](#))

B.2.2.5 diag::DoIPPowerMode::Offer function

[SWS_DM_00735]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::DoIPPowerMode	
Syntax:	<code>ara::core::Result<void> Offer ();</code>	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	<code>#include "ara/diag/doip_power_mode.h"</code>	
Description:	This Offer will enable the DM to forward request messages to this handler.	

]([RS_AP_00119](#), [RS_AP_00139](#), [SRS_Eth_00080](#))

B.2.2.6 diag::DoIPPowerMode::StopOffer function

[SWS_DM_00736]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::DoIPPowerMode
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/doip_power_mode.h"
Description:	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} [

Kind:	class
Symbol:	DoIPActivationLine
Scope:	namespace ara::diag
Syntax:	class DoIPActivationLine {...};
Header file:	#include "ara/diag/doip_activationline.h"
Description:	DiagnosticDoIPActivationLineInterface.

]([RS_Diag_04242](#))

B.2.3.1 diag::DoIPActivationLine::DoIPActivationLine function

[SWS_DM_00831]{DRAFT} [

Kind:	function	
Symbol:	DoIPActivationLine(const ara::core::InstanceSpecifier &specifier)	
Scope:	class ara::diag::DoIPActivationLine	
Syntax:	explicit DoIPActivationLine (const ara::core::InstanceSpecifier &specifier);	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of an DiagnosticDoIPActivationLineInterface
Header file:	#include "ara/diag/doip_activationline.h"	
Description:	Constructor of DoIPActivationLine.	

]([RS_Diag_04242](#))

B.2.3.2 diag::DoIPActivationLine::~DoIPActivationLine function

[SWS_DM_00832]{DRAFT} [

Kind:	function
Symbol:	~DoIPActivationLine()
Scope:	class ara::diag::DoIPActivationLine
Syntax:	virtual ~DoIPActivationLine () noexcept=default;
Exception Safety:	noexcept
Header file:	#include "ara/diag/doip_activationline.h"
Description:	Destructor of DoIPActivationLine.

]([RS_Diag_04242](#), [RS_AP_00134](#))

B.2.3.3 diag::DoIPActivationLine::GetNetworkInterfaceId function

[SWS_DM_00833]{DRAFT} [

Kind:	function
Symbol:	GetNetworkInterfaceId()
Scope:	class ara::diag::DoIPActivationLine
Syntax:	virtual ara::core::Future<std::uint8_t> GetNetworkInterfaceId ()=0;
Return value:	ara::core::Future< std::uint8_t > network interface id for which this activation line is responsible.
Header file:	#include "ara/diag/doip_activationline.h"
Description:	Called to get the network interface Id (see DolpNetworkConfiguration.networkInterfaceId) for which this DoIPActivationLine instance is responsible.
Notes:	If the reported DolpNetworkConfiguration.networkInterfaceId belongs to a DolpNetwork Configuration with property isActivationLineDependent = 'FALSE', this is an error!

]([RS_Diag_04242](#))

B.2.3.4 diag::DoIPActivationLine::UpdateActivationLineState function

[SWS_DM_00834]{DRAFT} [

Kind:	function
Symbol:	UpdateActivationLineState(bool)
Scope:	class ara::diag::DoIPActivationLine
Syntax:	virtual void UpdateActivationLineState (bool)=0;
DIRECTION NOT DEFINED	bool -
Return value:	None
Header file:	#include "ara/diag/doip_activationline.h"





Description:	Called to update current activation line state.
---------------------	---

](RS_Diag_04242)

B.2.3.5 diag::DoIPActivationLine::GetActivationLineState function

[SWS_DM_00835]{DRAFT} [

Kind:	function	
Symbol:	GetActivationLineState()	
Scope:	class ara::diag::DoIPActivationLine	
Syntax:	virtual ara::core::Future<bool> GetActivationLineState ()=0;	
Return value:	ara::core::Future< bool >	TRUE in case the activation line is active, else FALSE.
Header file:	#include "ara/diag/doip_activationline.h"	
Description:	Called to get the current activation line state.	

](RS_Diag_04242)

B.2.3.6 diag::DoIPActivationLine::Offer function

[SWS_DM_00836]{DRAFT} [

Kind:	function	
Symbol:	Offer()	
Scope:	class ara::diag::DoIPActivationLine	
Syntax:	ara::core::Result<void> Offer ();	
Return value:	ara::core::Result< void >	–
Errors:	DiagErrorDomain::DiagReporting Errc::kGenericError	General error occurred.
	DiagErrorDomain::DiagOfferErrc::k AlreadyOffered	This service was already offered.
Header file:	#include "ara/diag/doip_activationline.h"	
Description:	This Offer will enable the DM to listen to activation line state changes for the given interface.	

](RS_Diag_04242, RS_AP_00119)

B.2.3.7 diag::DoIPActivationLine::StopOffer function

[SWS_DM_00837]{DRAFT} [

Kind:	function
Symbol:	StopOffer()
Scope:	class ara::diag::DoIPActivationLine
Syntax:	void StopOffer ();
Return value:	None
Header file:	#include "ara/diag/doip_activationline.h"
Description:	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} [

Kind:	class
Symbol:	DoIPTriggerVehicleAnnouncement
Scope:	namespace ara::diag
Syntax:	class DoIPTriggerVehicleAnnouncement {...};
Header file:	#include "ara/diag/doip_trigger_announcement.h"
Description:	DiagnosticDoIPTriggerVehicleAnnouncement.

|(RS_Diag_04242)

B.2.4.1 diag::DoIPTriggerVehicleAnnouncement::GetDoIPTriggerVehicleAnnouncement function

[SWS_DM_00821]{DRAFT} [

Kind:	function
Symbol:	GetDoIPTriggerVehicleAnnouncement()
Scope:	class ara::diag::DoIPTriggerVehicleAnnouncement
Syntax:	static ara::core::Result<DoIPTriggerVehicleAnnouncement&> GetDoIPTriggerVehicleAnnouncement ();
Return value:	ara::core::Result< DoIPTriggerVehicleAnnouncement & > DoIPTriggerVehicleAnnouncement object
Header file:	#include "ara/diag/doip_trigger_announcement.h"
Description:	Get DoIPTriggerVehicleAnnouncement interface from DM.

|(RS_Diag_04242)

B.2.4.2 diag::DoIPTriggerVehicleAnnouncement::TriggerVehicleAnnouncement function

[SWS_DM_00822]{DRAFT} [

Kind:	function	
Symbol:	TriggerVehicleAnnouncement(std::uint8_t networkInterfaceId)	
Scope:	class ara::diag::DoIPTriggerVehicleAnnouncement	
Syntax:	virtual ara::core::Result<void> TriggerVehicleAnnouncement (std::uint8_t networkInterfaceId)=0;	
DIRECTION NOT DEFINED	networkInterfaceId	–
Return value:	ara::core::Result< void >	–
Header file:	#include "ara/diag/doip_trigger_announcement.h"	
Description:	Called by application to trigger DM sending out vehicle announcements on the given network interface Id.	
Notes:	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](#))

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 Traceables 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> <code>Address</code> type
[SWS_DM_00294]	meta info map type
[SWS_DM_00295]	meta info map vendor type
[SWS_DM_00296]	<code>TargetAddressType</code> <code>Address</code> 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





Number	Heading
[SWS_DM_00302]	AddMetaInfo method
[SWS_DM_00303]	UdsMessage Pointer
[SWS_DM_00304]	Const UdsMessage Pointer
[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





Number	Heading
[SWS_DM_00341]	Confirmation of service processing
[SWS_DM_00342]	Indication of UDS message reception
[SWS_DM_00343]	Acceptance of UDS message reception
[SWS_DM_00344]	Refusal of UDS message reception
[SWS_DM_00345]	Forwarding of UDS message
[SWS_DM_00346]	Aborting of UDS message
[SWS_DM_00347]	Channel identification in Indication
[SWS_DM_00348]	Transmission of UDS response message
[SWS_DM_00349]	Reuse channel identifier of Indication
[SWS_DM_00350]	Confirmation of UDS message transmission
[SWS_DM_00351]	Confirmation Result
[SWS_DM_00356]	Requesting Notification of a channel reestablishment
[SWS_DM_00357]	Validity/lifetime of a Notification Request
[SWS_DM_00358]	Notification of a channel reestablishment
[SWS_DM_00359]	Persistent Storage of Notification Request
[SWS_DM_00360]	EcuReset positive response processing after reset
[SWS_DM_00361]	EcuReset application error processing
[SWS_DM_00362]	Checking Supported Subfunction for CompareKey
[SWS_DM_00363]	Positive response processing
[SWS_DM_00364]	Negative response processing
[SWS_DM_00365]	Suppression of response
[SWS_DM_00366]	Suppression of response for functional requests
[SWS_DM_00367]	No service processing
[SWS_DM_00368]	Sending busy responses
[SWS_DM_00369]	Max. number of busy responses
[SWS_DM_00370]	Support of UDS service ReadDTCInformation, Subfunction 0x06
[SWS_DM_00371]	Support of UDS service ReadDTCInformation, Subfunction 0x14
[SWS_DM_00372]	Support of UDS service ReadDTCInformation, Subfunction 0x17
[SWS_DM_00373]	Support of UDS service ReadDTCInformation, Subfunction 0x18
[SWS_DM_00374]	Support of UDS service ReadDTCInformation, Subfunction 0x19

Table D.1: Added Traceables in 17-10

D.1.2 Changed Traceables 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
[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



△

Number	Heading
[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
[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 Traceables in 17-10

D.1.3 Deleted Traceables in 17-10

Number	Heading
[SWS_DM_00001]	Availability of operation cycle service interfaces
[SWS_DM_00053]	Cancellation of <code>Active Protocol</code>
[SWS_DM_00054]	Generic UDS Service Interface
[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 Traceables 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 Traceables 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
[SWS_DM_00413]	Check supported DataIdentifier in active session





Number	Heading
[SWS_DM_00414]	Check supported DataIdentifier on active security level
[SWS_DM_00415]	Check supported DataIdentifier
[SWS_DM_00416]	Check supported DataIdentifier in active session
[SWS_DM_00417]	Check supported DataIdentifier on active security level
[SWS_DM_00418]	Writing data for requested DataIdentifier
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00420]	Instantiation of Diagnostic Server
[SWS_DM_00434]	Providing the PowerMode in DoIP protocol messages
[SWS_DM_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 Traceables in 18-03

D.2.2 Changed Traceables in 18-03

Number	Heading
[SWS_DM_00002]	Automatic starting of operation cycles
[SWS_DM_00003]	Automatic ending of operation cycles
[SWS_DM_00005]	DoIP Support
[SWS_DM_00007]	Uniqueness of diagnostic events
[SWS_DM_00008]	Diagnostic event processing interface
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00013]	Events without debouncing
[SWS_DM_00014]	Use of counter-based debouncing for events
[SWS_DM_00015]	Use of timer based debouncing for events
[SWS_DM_00017]	Calculation of the FDC based on the internal debounce counter
[SWS_DM_00018]	Internal debounce counter init and storage
[SWS_DM_00019]	Internal debounce counter incrementation
[SWS_DM_00020]	Internal debounce counter decrementation
[SWS_DM_00021]	Direct failed qualification of counter-based events
[SWS_DM_00022]	Debounce counter jump up behavior
[SWS_DM_00023]	Debounce counter jump down behavior
[SWS_DM_00024]	Qualified failed event using counter-based debouncing
[SWS_DM_00025]	Qualified passed event using counter-based debouncing





Number	Heading
[SWS_DM_00026]	Application resetting the debounce counter
[SWS_DM_00028]	Debounce counter persistency
[SWS_DM_00029]	Direct passed qualification of counter-based events
[SWS_DM_00030]	Calculation of the FDC based on the internal debounce timer
[SWS_DM_00031]	Starting time-based event debouncing for failed
[SWS_DM_00032]	Restrictions on restarting a running event debounce timer for failed
[SWS_DM_00033]	Debounce timer behavior upon reported failed
[SWS_DM_00034]	Starting time-based event debouncing for passed
[SWS_DM_00035]	Restrictions on restarting a running event debounce timer for passed
[SWS_DM_00036]	Debounce timer behavior upon reported passed
[SWS_DM_00037]	Debounce time freeze request
[SWS_DM_00038]	Continuing a frozen debounce timer
[SWS_DM_00039]	Resetting the debounce counter upon starting or restarting an operation cycle
[SWS_DM_00040]	Definition of debounce counter reset
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00042]	Cancelling external service processors
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00046]	Each Diagnostic Protocol has own session resources
[SWS_DM_00047]	Each Diagnostic Protocol has own security-level resources
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00049]	Refusal of second diagnostic request from different diagnostic client with BusyRepeatRequest
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00055]	Supported event memories
[SWS_DM_00057]	Availability of a user-defined event memory
[SWS_DM_00058]	DTC interpretation format
[SWS_DM_00060]	Set of supported DTCs
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[1] 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
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason





Number	Heading
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00085]	Internal debounce counter init
[SWS_DM_00086]	Resetting the debounce counter after clearing DTC
[SWS_DM_00087]	Enable condition influence on debouncing behavior (freeze)
[SWS_DM_00088]	ControlDTCSetting influence (freeze)
[SWS_DM_00089]	Reporting PREPASSED or PREFAILED for events without assigned debouncing algorithm
[SWS_DM_00090]	Support of UDS service ClearDiagnosticInformation
[SWS_DM_00091]	Evaluation of ClearDiagnosticInformation parameters
[SWS_DM_00092]	Parameter range check for groupOfDTC request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00097]	Abort on failed verification step
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service 0x14
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00116]	Clearing a DTC group
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00118]	Event specific configuration to allow clearing of a DTC
[SWS_DM_00119]	Init value for events with clear allowed information
[SWS_DM_00120]	Description of application interface to control the clear event behavior
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block status byte clearing during a clear DTC operation
[SWS_DM_00124]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[SWS_DM_00128]	Realisation of UDS service 0x34 RequestDownload
[SWS_DM_00129]	Supported addressAndLengthFormatIdentifier
[SWS_DM_00130]	Not supported addressAndLengthFormatIdentifier
[SWS_DM_00136]	Request upload service processing
[SWS_DM_00138]	Transfer data service processing
[SWS_DM_00139]	Transfer data service validation
[SWS_DM_00142]	Transfer data service processing
[SWS_DM_00143]	Transfer data service validation
[SWS_DM_00144]	Parallel clearing DTCs in different DiagnosticMemoryDestination
[SWS_DM_00145]	Allow only one simultaneous clear DTC operation for one DiagnosticMemoryDestination





Number	Heading
[SWS_DM_00146]	Unlock clear DTC operation for one DiagnosticMemoryDestination
[SWS_DM_00147]	Behavior while trying to clear DTCs on a locked DiagnosticMemoryDestination
[SWS_DM_00148]	Persistent storage of event memory entries
[SWS_DM_00151]	Snapshot record numeration
[SWS_DM_00152]	Number of snapshot records for a DTC
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00154]	Number of extended data for a DTC
[SWS_DM_00155]	Extended data record numeration
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00159]	Allow only to clear GroupOfAllDTCs
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00161]	Negative response on not supported GroupOfDTC parameter
[SWS_DM_00162]	Point in time for positive response for ClearDTC
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of DiagnosticMonitor service interfaces
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnosticInformation
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00205]	Providing the VIN in DoIP protocol messages
[SWS_DM_00213]	DTC status processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00217]	DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	Confirmation
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about the changes of the status byte
[SWS_DM_00223]	Handling of 'warningIndicatorRequested' bit
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00229]	Support of UDS service ControlDTCSetting
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter





Number	Heading
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00238]	Aging and healing
[SWS_DM_00239]	Aging counter
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Reoccurrence after aging
[SWS_DM_00243]	Aging-related UDS status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00258]	Cancellation of <code>Active Protocol</code> in non-default session
[SWS_DM_00259]	Completion of already <code>Active Protocols</code> in default session
[SWS_DM_00260]	instances of interface <code>ClearDTC</code>
[SWS_DM_00261]	Usage of <code>ClearDTC</code> Interface
[SWS_DM_00262]	Common semantic behavior for <code>ClearDTC</code> triggered via diagnostics or application
[SWS_DM_00265]	<code>ClearDTC</code> called while another clear operation is in progress
[SWS_DM_00268]	<code>EcuReset</code> positive response processing before reset
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00272]	Expiration of the delay timer
[SWS_DM_00273]	Notification event upon <code>snapshot record</code> updates
[SWS_DM_00277]	Cancellation of <code>Active Protocol</code> in case of External Service Processing
[SWS_DM_00278]	Cancellation of <code>Active Protocol</code> in case of Internal Processing
[SWS_DM_00279]	Cancellation of <code>Active Protocol</code> before Response Transmission
[SWS_DM_00280]	Cancellation of <code>Active Protocol</code> at Response Transmission
[SWS_DM_00281]	Cancellation of active <code>DiagnosticConversation</code> in Non-Default Session
[SWS_DM_00282]	Handling of non-/active diagnostic conversations
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00290]	Refusal of second diagnostic request from different diagnostic client without response
[SWS_DM_00309]	<code>IndicateMessage</code> method





Number	Heading
[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
[SWS_DM_CON-STR_00168]	Required operation cycles for diagnostic events
[SWS_DM_CON-STR_00206]	Supported format for data identifier for VINDataIdentifier





Number	Heading
[SWS_DM_CON-STR_00207]	Required VINDataIdentifier

Table D.5: Changed Traceables in 18-03

D.2.3 Deleted Traceables 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 DTCTOrigin
[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 Traceables 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 Traceables in 18-10

Number	Heading
[SWS_DM_00421]	Identification of a Diagnostic Client
[SWS_DM_00422]	Instantiation of Diagnostic Conversation Service Interface
[SWS_DM_00423]	Assignment of Diagnostic Conversation to Service Instances
[SWS_DM_00424]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00425]	Procedure to assign UDS requests to Diagnostic Conversations
[SWS_DM_00426]	Assigning a UDS request to an existing Diagnostic Conversation
[SWS_DM_00427]	Priority of a Diagnostic Conversation
[SWS_DM_00428]	Treatment of priority values
[SWS_DM_00429]	Prioritization in case of Pseudo Parallel Mode and active non-default session
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations
[SWS_DM_00431]	Replacement of Diagnostic Conversations
[SWS_DM_00432]	Initial values for Diagnostic Conversation
[SWS_DM_00433]	Refusal of diagnostic request due to busy Diagnostic Conversation
[SWS_DM_00435]	Default session change trigger from AAs
[SWS_DM_00436]	Providing the GID in DoIP protocol messages
[SWS_DM_00437]	Check supported RoutineIdentifier on active security level
[SWS_DM_00438]	Check Support of UDS service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of UDS service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of UDS service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of UDS service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of UDS service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of UDS service RequestTransferExit (0x37) on active security level
[SWS_DM_00444]	Check Support of UDS service ControlDTCSetting (0x85) in active session
[SWS_DM_00445]	Check Support of UDS service ControlDTCSetting (0x85) on active security level
[SWS_DM_00446]	Check Support of UDS service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of UDS service RequestDownload (0x34) on active security level





Number	Heading
[SWS_DM_00448]	Check supported RoutineIdentifier in active session
[SWS_DM_00449]	Supported DoIP message types
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00475]	DoIP Version
[SWS_DM_00476]	User Controlled Warning IndicatorRequest-bit
[SWS_DM_00477]	Not Storing of 'warningIndicatorRequested' bit
[SWS_DM_00478]	Persistent Storage of failed attempts to change security level
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00480]	Security Access Blocking Timer
[SWS_DM_00481]	Handling of DiagnosticClearConditions
[SWS_DM_00482]	Cancellation of a Diagnostic Conversation
[SWS_DM_00483]	Cancellation trigger from AAs
[SWS_DM_00484]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00485]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00487]	Ignoring UDS message reception because of unknown target address
[SWS_DM_00491]	Realisation of UDS service 0x86 ResponseOnEvent
[SWS_DM_00492]	Client Server communication
[SWS_DM_00493]	Reestablishing of Client Server communication
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00495]	Start initialisation of ResponseOnEvent
[SWS_DM_00496]	Stop initialisation of ResponseOnEvent
[SWS_DM_00497]	Clear initialisation of ResponseOnEvent
[SWS_DM_00498]	Exclusive ResponseOnEvent 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
[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]	





Number	Heading
[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 Traceables in 18-10

D.3.2 Changed Traceables 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 <i>Diagnostic Conversation</i> has its own session resources
[SWS_DM_00047]	Each <i>Diagnostic Conversation</i> has its own security-level resources
[SWS_DM_00049]	Refusal of diagnostic request due to prioritization with BusyRepeatRequest
[SWS_DM_00061]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCByStatusMask
[SWS_DM_00062]	Mapping between ISO 14229-1[1] and Autosar Diagnostic Extract Template [3] of the DTCFormatIdentifier
[SWS_DM_00063]	Providing rule for DTCFormatIdentifier in positive response ReadDTCInformation.reportNumberOfDTCBySeverityMaskRecord





Number	Heading
[SWS_DM_00067]	Monitor initialization for clearing reason
[SWS_DM_00068]	Monitor initialization for operation cycle restart reason
[SWS_DM_00069]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00070]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00071]	Monitor initialization for storage condition reenabling reason
[SWS_DM_00074]	Handling of enable conditions
[SWS_DM_00089]	Reporting <code>kPrepassed</code> or <code>kPrefailed</code> for events without an assigned de-bouncing algorithm
[SWS_DM_00090]	Support of UDS service <code>ClearDiagnosticInformation</code>
[SWS_DM_00091]	Evaluation of <code>ClearDiagnosticInformation</code> parameters
[SWS_DM_00092]	Parameter range check for <code>groupOfDTC</code> request parameter
[SWS_DM_00096]	Validation Steps and Order
[SWS_DM_00098]	UDS message checks
[SWS_DM_00099]	Supported Service SID level checks
[SWS_DM_00100]	Supported Service subfunction level checks
[SWS_DM_00101]	Session Access SID level Permission
[SWS_DM_00102]	Session Access subfunction level Permission
[SWS_DM_00103]	Security Access level Permission
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00111]	Configurable environment condition checks
[SWS_DM_00112]	Condition check definition
[SWS_DM_00113]	Positive response for UDS service <code>0x14</code>
[SWS_DM_00114]	Limitation to one simultaneous DTC clear operation
[SWS_DM_00115]	Memory error handling while clearing DTCs
[SWS_DM_00117]	Clearing a DTC
[SWS_DM_00121]	Forbidden clearing of snapshot records and extended data records
[SWS_DM_00122]	UDS response behavior on not allowed clear operations
[SWS_DM_00123]	Block status byte clearing during a clear DTC operation
[SWS_DM_00124]	Limited status byte clearing during a clear DTC operation
[SWS_DM_00126]	Realisation of UDS service <code>0x3E</code> <code>TesterPresent</code>
[SWS_DM_00127]	Availability of diagnostic service processors
[SWS_DM_00128]	Realization of UDS service <code>RequestDownload</code> (<code>0x34</code>)
[SWS_DM_00129]	Supported <code>addressAndLengthFormatIdentifier</code>
[SWS_DM_00130]	Not supported <code>addressAndLengthFormatIdentifier</code>
[SWS_DM_00131]	UDS service <code>RequestDownload</code> (<code>0x34</code>) processing
[SWS_DM_00134]	Realization of UDS service <code>RequestUpload</code> (<code>0x35</code>)
[SWS_DM_00136]	UDS service <code>RequestUpload</code> (<code>0x35</code>) processing





Number	Heading
[SWS_DM_00137]	Realization of UDS service TransferData (0x36)
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00140]	Realisation of UDS service 0x28 CommunicationControl
[SWS_DM_00141]	Realization of UDS service RequestTransferExit (0x37)
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00159]	Allow only to clear GroupOfAllDTCs
[SWS_DM_00160]	Allow to clear single DTCs
[SWS_DM_00162]	Point in time for positive response for ClearDTC
[SWS_DM_00163]	Definition of a failed clear operation with event clear allowed and event combination
[SWS_DM_00164]	Definition of a failed clear operation with event clear allowed and clearing a group of DTCs
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of DiagnosticMonitor service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00170]	Realisation of UDS service ReadDataByIdentifier (0x22)
[SWS_DM_00177]	Reaction on ApplicationError
[SWS_DM_00186]	Realisation of UDS service WriteDataByIdentifier (0x2E)
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00193]	Support of a user-defined fault memory clear request
[SWS_DM_00194]	Definition of the user-defined fault memory number for ClearDiagnosticInformation
[SWS_DM_00195]	Clearing a user-defined memory
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00199]	Positive Response processing
[SWS_DM_00201]	Realization of UDS service RoutineControl (0x31)
[SWS_DM_00202]	Check for Supported RoutineIdentifier and Reaction
[SWS_DM_00203]	Check for Supported Subfunction and Reaction
[SWS_DM_00205]	Providing the VIN in DoIP protocol messages
[SWS_DM_00208]	Validation of the requested user-defined memory number
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00213]	DTC status processing





Number	Heading
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the DTC
[SWS_DM_00216]	DTC status bit transitions triggered by operation cycle changes
[SWS_DM_00217]	DTC status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_00218]	Confirmation
[SWS_DM_00219]	Observability of the status byte
[SWS_DM_00220]	Notification about DTC status changes
[SWS_DM_00222]	Observability of indicator status
[SWS_DM_00226]	Support of UDS service DiagnosticSessionControl
[SWS_DM_00227]	Check for supported sessions
[SWS_DM_00228]	Switch to requested Diagnostic Session
[SWS_DM_00229]	Support of UDS service ControlDTCSetting (0x85)
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00231]	Invalid value for optional request parameter
[SWS_DM_00232]	Support of Subfunction 0x01 (ON)
[SWS_DM_00233]	Support of Subfunction 0x02 (OFF)
[SWS_DM_00234]	Support of UDS service ECUReset
[SWS_DM_00235]	ECUReset service processing
[SWS_DM_00236]	Realization of UDS service 0x27 SecurityAccess
[SWS_DM_00237]	Aging
[SWS_DM_00240]	Processing the aging counter
[SWS_DM_00241]	Aging cycle and threshold
[SWS_DM_00242]	Re-occurrence after aging
[SWS_DM_00243]	Aging-related UDS DTC status byte processing
[SWS_DM_00244]	Support of UDS service ReadDTCInformation, Subfunction 0x01
[SWS_DM_00245]	Support of UDS service ReadDTCInformation, Subfunction 0x02
[SWS_DM_00246]	Support of UDS service ReadDTCInformation, Subfunction 0x04
[SWS_DM_00247]	Support of UDS service ReadDTCInformation, Subfunction 0x07
[SWS_DM_00248]	Notification about session change
[SWS_DM_00249]	Checking Supported Subfunction for RequestSeed
[SWS_DM_00250]	Notification about security-level change
[SWS_DM_00252]	Reaction on Unsupported Subfunction
[SWS_DM_00260]	instances of interface ClearDTC
[SWS_DM_00261]	Usage of ClearDTC Interface
[SWS_DM_00263]	ClearDTC call on invalid DTC or DTCgroup
[SWS_DM_00265]	ClearDTC called while another clear operation is in progress
[SWS_DM_00266]	ClearDTC processing in case of memory errors
[SWS_DM_00267]	Possible failure of ClearDTC





Number	Heading
[SWS_DM_00268]	EcuReset positive response processing before reset
[SWS_DM_00269]	Reaction on Unsupported Subfunction
[SWS_DM_00270]	Counting of attempts to change security level
[SWS_DM_00271]	Evaluate the number of failed security level change attempts
[SWS_DM_00273]	Notification event upon <code>snapshot record</code> updates
[SWS_DM_00277]	Cancellation of a Diagnostic Conversation in case of External Service Processing
[SWS_DM_00278]	Cancellation of a Diagnostic Conversation in case of Internal Processing
[SWS_DM_00279]	Cancellation of a Diagnostic Conversation before Response Transmission
[SWS_DM_00280]	Cancellation of a Diagnostic Conversation at Response Transmission
[SWS_DM_00286]	Configurable environmental condition check execution
[SWS_DM_00288]	Configurable environmental condition check evaluates to <code>TRUE</code>
[SWS_DM_00289]	Configurable environmental condition check evaluates to <code>FALSE</code>
[SWS_DM_00290]	Refusal of diagnostic request due to prioritization without response
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	
[SWS_DM_00298]	
[SWS_DM_00299]	
[SWS_DM_00300]	
[SWS_DM_00301]	
[SWS_DM_00302]	
[SWS_DM_00303]	
[SWS_DM_00304]	
[SWS_DM_00306]	
[SWS_DM_00307]	
[SWS_DM_00309]	
[SWS_DM_00310]	
[SWS_DM_00311]	
[SWS_DM_00312]	
[SWS_DM_00313]	
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	





Number	Heading
[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[1]
[SWS_DM_00366]	Suppression of negative response for functional requests in accordance to ISO 14229-1[1]
[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>
[SWS_DM_00407]	Default Service Interface for writing <code>DiagnosticDataIdentifier</code>
[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





Number	Heading
[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 <code>PowerMode</code> in DoIP protocol messages

Table D.8: Changed Traceables in 18-10

D.3.3 Deleted Traceables in 18-10

Number	Heading
[SWS_DM_00001]	SRS Diagnostics
[SWS_DM_00012]	DoIP configurable source address identification
[SWS_DM_00041]	Behavior according to ISO Multiple client handling flow
[SWS_DM_00043]	Request refusal in case of no resources
[SWS_DM_00044]	Request refusal in case of non-default session active
[SWS_DM_00045]	Ignore ISO same resource access check
[SWS_DM_00048]	Request refusal in case of no resources
[SWS_DM_00051]	Cancellation of Active Protocol with lower priority
[SWS_DM_00052]	Selection between multiple cancellation candidates
[SWS_DM_00066]	Monitor initialization
[SWS_DM_00105]	Configurable Manufacturer Permission Check Services
[SWS_DM_00107]	Configurable Supplier Permission Check Services
[SWS_DM_00118]	Event specific configuration to allow clearing of a DTC
[SWS_DM_00119]	Init value for events with clear allowed information
[SWS_DM_00120]	Description of application interface to control the clear event behavior
[SWS_DM_00125]	Linking between event clear allowed and clearing a DTC
[SWS_DM_00161]	Negative response on not supported GroupOfDTC parameter
[SWS_DM_00166]	Trigger to process event status
[SWS_DM_00180]	Provide Protocol Priority Configurability
[SWS_DM_00182]	Identification of a protocol for Priority Assignment
[SWS_DM_00183]	Wildcards per attribute
[SWS_DM_00184]	Protocol Match Search
[SWS_DM_00185]	No Match
[SWS_DM_00258]	Cancellation of Active Protocol in non-default session
[SWS_DM_00259]	Completion of already Active Protocols in default session
[SWS_DM_00274]	Definition of an active <code>Diagnostic Protocol</code>
[SWS_DM_00281]	Cancellation of active <code>DiagnosticConversation</code> in Non-Default Session





Number	Heading
[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 Traceables 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 Traceables 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
[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





Number	Heading
[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]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	





Number	Heading
[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]	
[SWS_DM_00658]	
[SWS_DM_00663]	
[SWS_DM_00664]	
[SWS_DM_00665]	





Number	Heading
[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]	
[SWS_DM_00733]	
[SWS_DM_00734]	
[SWS_DM_00735]	
[SWS_DM_00736]	



△

Number	Heading
[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 Traceables in 19-03

D.4.2 Changed Traceables 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[1]
[SWS_DM_00367]	No service processing
[SWS_DM_00376]	Positive response processing
[SWS_DM_00382]	Session timeout start
[SWS_DM_00383]	Session timeout stop
[SWS_DM_00384]	
[SWS_DM_00419]	Reaction on ApplicationError
[SWS_DM_00436]	Providing the GID in DoIP protocol messages
[SWS_DM_00479]	Blocking Timer for security access on Restart or Power down - power up cycle
[SWS_DM_00503]	Reading Diagnostic Data Identifier by DataIdentifier interface





Number	Heading
[SWS_DM_00504]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00651]	
[SWS_DM_09017]	
[SWS_DM_CON-STR_00395]	Restriction on DEM-exclusive DiagnosticDataElements

Table D.11: Changed Traceables in 19-03

D.4.3 Deleted Traceables in 19-03

Number	Heading
[SWS_DM_00104]	Supported UDS Services
[SWS_DM_00483]	Cancellation trigger from AAs
[SWS_DM_09028]	

Table D.12: Deleted Traceables 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 Traceables 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]	
[SWS_DM_00809]	





Number	Heading
[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>AAs</code>
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00844]	Updating DiagnosticConversation Service Instance fields
[SWS_DM_00845]	Notification about session change
[SWS_DM_00846]	Notification about security-level change
[SWS_DM_00847]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00849]	Reading Diagnostic Data Identifier by GenericUDSService interface
[SWS_DM_00850]	Default Service Interface for reading <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[1]
[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 <code>DTC status bit</code> transitions triggered by test results
[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_00886]	Observability of the status byte
[SWS_DM_00887]	Notification about <code>DTC</code> 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 <code>snapshot record</code> storage
[SWS_DM_00894]	Notification event upon <code>snapshot record</code> updates
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00896]	Handling of <code>DiagnosticClearConditions</code>
[SWS_DM_00897]	Usage of ClearDTC Interface





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 Traceables in 19-11

D.5.2 Changed Traceables 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 ClearDiagnosticInformation <code>UDS service</code>
[SWS_DM_00218]	Trip Counter
[SWS_DM_00242]	Re-occurrence after <code>Aging</code>
[SWS_DM_00243]	<code>Aging</code> -related <code>UDS DTC status byte</code> processing





Number	Heading
[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 <i>internal DiagnosticDataElements</i>
[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]	
[SWS_DM_00553]	





Number	Heading
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00571]	Reaction on ApplicationError
[SWS_DM_00572]	Writing data for requested DataIdentifier
[SWS_DM_00573]	Reaction on ApplicationError
[SWS_DM_00574]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00575]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00576]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00584]	
[SWS_DM_00585]	
[SWS_DM_00586]	
[SWS_DM_00587]	
[SWS_DM_00588]	
[SWS_DM_00589]	
[SWS_DM_00590]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00616]	





Number	Heading
[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]	
[SWS_DM_00698]	





Number	Heading
[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]	
[SWS_DM_00790]	





Number	Heading
[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 Traceables in 19-11

D.5.3 Deleted Traceables 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)
[SWS_DM_00089]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00106]	Signature of Manufacturer Permission Check Method





Number	Heading
[SWS_DM_00108]	Signature of Supplier Permission Check Method
[SWS_DM_00131]	UDS service RequestDownload (0x34) processing
[SWS_DM_00136]	UDS service RequestUpload (0x35) processing
[SWS_DM_00138]	UDS service TransferData (0x36) processing
[SWS_DM_00139]	UDS service TransferData (0x36) validation
[SWS_DM_00142]	UDS service RequestTransferExit (0x37) processing
[SWS_DM_00143]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00153]	Triggering for snapshot record storage
[SWS_DM_00156]	Triggering for extended data record storage and updates
[SWS_DM_00167]	Ignoring reported events for not started operation cycles
[SWS_DM_00168]	Availability of <code>DiagnosticMonitor</code> service interfaces
[SWS_DM_00169]	Restart of operation cycles
[SWS_DM_00177]	Reaction on <code>ApplicationError</code>
[SWS_DM_00192]	Operation cycles are only ended once
[SWS_DM_00197]	Communication control service processing
[SWS_DM_00198]	Negative Response processing
[SWS_DM_00205]	Providing the <code>VIN</code> in DoIP protocol messages
[SWS_DM_00210]	UDS Service RoutineControl (0x31) startRoutine processing
[SWS_DM_00211]	UDS Service RoutineControl (0x31) requestRoutineResults processing
[SWS_DM_00212]	UDS Service RoutineControl (0x31) stopRoutine processing
[SWS_DM_00214]	DTC status bit transitions triggered by test results
[SWS_DM_00215]	Resetting the status of the <code>DTC</code>
[SWS_DM_00216]	<code>DTC</code> 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
[SWS_DM_00267]	Possible failure of ClearDTC
[SWS_DM_00273]	Notification event upon <code>snapshot record</code> updates





Number	Heading
[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[1]
[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 <code>DataIdentifier</code>
[SWS_DM_00418]	Writing data for requested <code>DataIdentifier</code>
[SWS_DM_00419]	Reaction on <code>ApplicationError</code>
[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 <code>DiagnosticConversation</code> Service Instance fields
[SWS_DM_00485]	Reinitialization of Service Instance on Cancellation of a Diagnostic Conversation
[SWS_DM_00503]	Reading Diagnostic Data Identifier by <code>DataIdentifier</code> interface
[SWS_DM_00504]	Reading Diagnostic Data Identifier by <code>GenericUDSService</code> interface
[SWS_DM_00505]	Writing Diagnostic Data Identifier by <code>DataIdentifier</code> interface
[SWS_DM_00506]	Writing Diagnostic Data Identifier by <code>GenericUDSService</code> interface
[SWS_DM_00508]	Reading <code>DiagnosticDataIdentifier</code> configured for representing <code>VIN</code>
[SWS_DM_00509]	Writing <code>DiagnosticDataIdentifier</code> 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]	<code>NumberOfStoredEntries</code>

Table D.15: Deleted Traceables 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 Traceables 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]	





Number	Heading
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00621]	
[SWS_DM_00622]	
[SWS_DM_00623]	
[SWS_DM_00624]	
[SWS_DM_00625]	
[SWS_DM_00626]	
[SWS_DM_00627]	
[SWS_DM_00628]	
[SWS_DM_00629]	
[SWS_DM_00630]	
[SWS_DM_00631]	
[SWS_DM_00632]	
[SWS_DM_00633]	
[SWS_DM_00705]	
[SWS_DM_00706]	
[SWS_DM_00707]	
[SWS_DM_00708]	
[SWS_DM_00786]	
[SWS_DM_00916]	Priority values
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00920]	Configuration of the event memory size
[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





Number	Heading
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00945]	Occurrence Counter initial value
[SWS_DM_00946]	Occurrence Counter increment strategy 'testFailed'-only
[SWS_DM_00947]	Occurrence Counter increment strategy 'confirmedDtcBit'
[SWS_DM_00948]	Occurrence Counter upper limit
[SWS_DM_00949]	Generation and usage of internal DiagnosticDataElements
[SWS_DM_00950]	Configuration of DTC priority as extended data record
[SWS_DM_00951]	Configuration of DTC "current FDC" as extended data record
[SWS_DM_00952]	Configuration of DTC "max. FDC since clear" as extended data record
[SWS_DM_00953]	Configuration of DTC "max. FDC current cycle" as extended data record
[SWS_DM_00954]	Configuration of DTC "occurrence counter" as extended data record
[SWS_DM_00955]	Configuration of DTC "aging counter up/down" as extended data record
[SWS_DM_00956]	Configuration of DTC "aging counter up" as extended data record
[SWS_DM_00957]	Configuration of DTC "aging counter down" as extended data record
[SWS_DM_00958]	Default value for DTC "aging counter up" if aging is not allowed
[SWS_DM_00959]	Default value for DTC "aging counter down" if aging is not allowed
[SWS_DM_00961]	Configuration of a DTCs significance as extended data record
[SWS_DM_00962]	Configuration of a DTCs Failed Operation Cycles as extended data record
[SWS_DM_00963]	Configuration of a DTCs failed operation Cycles Since First Failed as extended data record
[SWS_DM_00964]	Configuration of a DTCs failed operation Cycles Since Last Failed as extended data record
[SWS_DM_00965]	Caching of monitor results
[SWS_DM_00966]	Reporting of DTCStatusAvailabilityMask
[SWS_DM_00967]	Support of UDS service ReadDTCInformation, Subfunction 0x0A
[SWS_DM_00968]	Reporting of DTCAndStatusRecord parameter
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00970]	Behavior of failed data element retrieval
[SWS_DM_00971]	
[SWS_DM_00972]	





Number	Heading
[SWS_DM_00973]	
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00981]	Conditions of status based reporting order
[SWS_DM_00982]	Reporting order direction
[SWS_DM_00983]	Processing of Custom Diagnostic Services
[SWS_DM_00984]	Return of cancellation status
[SWS_DM_00989]	
[SWS_DM_00990]	
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[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]	





Number	Heading
[SWS_DM_01018]	ECUReset ara::diag::ResetRequestType check
[SWS_DM_01019]	Custom ara::diag::ResetRequestType processing
[SWS_DM_01020]	EnableRapidPowerShutdown processing
[SWS_DM_01021]	DisableRapidPowerShutdown processing
[SWS_DM_01022]	Block requests after ara::diag::EcuResetRequest::RequestReset called
[SWS_DM_01023]	Positive response before reset assurance
[SWS_DM_01024]	Event Status processing
[SWS_DM_01025]	Event status bit transitions triggered by test results
[SWS_DM_01026]	Resetting the status of an Event
[SWS_DM_01027]	Event status bit transitions triggered by operation cycle changes
[SWS_DM_01028]	Event status bit transitions triggered by ClearDiagnosticInformation UDS service
[SWS_DM_01029]	Notification about Event status changes
[SWS_DM_01030]	Observability of the UDS DTC status byte
[SWS_DM_01031]	Notification about UDS DTC status changes
[SWS_DM_01032]	Handling of 'WIR' bit without connected indicators
[SWS_DM_01033]	User controlled set of WIR-bit
[SWS_DM_01034]	User controlled reset of WIR-bit
[SWS_DM_01035]	User controlled WIR-bit handling and ControlDTCSetting
[SWS_DM_01037]	Behavior of not configured DiagnosticEvent.confirmationThreshold
[SWS_DM_01038]	Reading Diagnostic Data Identifier by ara::diag::GenericDataIdentifier interface
[SWS_DM_01039]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_01040]	Realization of UDS service ReadDataByPeriodicIdentifier(0x2A)
[SWS_DM_01041]	Check requested number of periodic DataIdentifiers
[SWS_DM_01042]	Minimum length check for ReadDataByPeriodicIdentifier
[SWS_DM_01043]	Check supported periodic DataIdentifier
[SWS_DM_01044]	Check Transmission Mode
[SWS_DM_01045]	Check Scheduler Availability
[SWS_DM_01046]	Check supported DataIdentifier in active session
[SWS_DM_01047]	Check supported DataIdentifier on active security level
[SWS_DM_01048]	Check DataIdentifier for environmental conditions
[SWS_DM_01049]	Checks Dynamically Defined DIDs in ReadDataByPeriodicIdentifier
[SWS_DM_01050]	Periodic DID length check
[SWS_DM_01051]	DM behavior on transmission Mode stopSending without periodicDataIdentifier in the request
[SWS_DM_01052]	DM behavior on transmission Mode stopSending with supported periodicDataIdentifier in the request





Number	Heading
[SWS_DM_01053]	DM behavior on transmission Mode stopSending with not supported periodicDataIdentifier in the request
[SWS_DM_01054]	Starting to transmit PDIDs after positive response
[SWS_DM_01055]	Reaction on ApplicationError
[SWS_DM_01056]	Optional condition checks for sending periodic 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 Traceables in R20-11

D.6.2 Changed Traceables 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[1] 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]	
[SWS_DM_00557]	





Number	Heading
[SWS_DM_00559]	
[SWS_DM_00560]	
[SWS_DM_00562]	Monitor initialization for clearing reason
[SWS_DM_00563]	Monitor initialization for operation cycle restart reason
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00568]	Handling of enable conditions
[SWS_DM_00570]	Retrieving data for requested DataIdentifier
[SWS_DM_00577]	Canceling external service processors
[SWS_DM_00585]	
[SWS_DM_00587]	
[SWS_DM_00589]	
[SWS_DM_00591]	
[SWS_DM_00592]	
[SWS_DM_00593]	
[SWS_DM_00594]	
[SWS_DM_00596]	
[SWS_DM_00597]	
[SWS_DM_00598]	
[SWS_DM_00599]	
[SWS_DM_00601]	
[SWS_DM_00602]	
[SWS_DM_00603]	
[SWS_DM_00604]	
[SWS_DM_00605]	
[SWS_DM_00607]	
[SWS_DM_00608]	
[SWS_DM_00609]	
[SWS_DM_00610]	
[SWS_DM_00611]	
[SWS_DM_00612]	
[SWS_DM_00613]	
[SWS_DM_00614]	
[SWS_DM_00615]	
[SWS_DM_00616]	
[SWS_DM_00618]	
[SWS_DM_00619]	
[SWS_DM_00634]	





Number	Heading
[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]	
[SWS_DM_00836]	





Number	Heading
[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 <i>DTC status bit</i> transitions triggered by test results
[SWS_DM_00895]	Triggering for extended data record storage and updates
[SWS_DM_00898]	ClearDTC call on invalid <i>DTC</i> or <i>DTC group</i>
[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 <i>DTC</i> format
[SWS_DM_CON-STR_00396]	Restriction on DCM-exclusive <i>DiagnosticDataElements</i>

Table D.17: Changed Traceables in R20-11

D.6.3 Deleted Traceables 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 <i>DTC</i>
[SWS_DM_00885]	UDS <i>DTC status bit</i> transitions triggered by <i>operation cycle</i> changes
[SWS_DM_00887]	Notification about <i>DTC</i> status changes

Table D.18: Deleted Traceables 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 Traceables 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





Number	Heading
[SWS_DM_01101]	Debouncing parameters from Monitor Constructor
[SWS_DM_01102]	
[SWS_DM_01103]	Caching of RestartOperationCycle
[SWS_DM_01104]	Operation Cycle restart
[SWS_DM_01105]	Restart OperationCycle during the processing of previous call
[SWS_DM_01106]	Applicability of Event Combination
[SWS_DM_01107]	DTC Status Byte calculation
[SWS_DM_01108]	Clear all DTCs event with event combination
[SWS_DM_01109]	UDS DTC status update for combined DTCs
[SWS_DM_01110]	Callbacks for combined UDS DTC status change
[SWS_DM_01111]	Fault detection counter for combined events
[SWS_DM_01112]	Event memory entry for events with the combination on storage
[SWS_DM_01113]	Aging counter for combined events
[SWS_DM_01114]	Data storage for event combination on retrieval
[SWS_DM_01115]	Data reporting for event combination on retrieval
[SWS_DM_01116]	Reporting order of snapshot and extended data records
[SWS_DM_01117]	Support of eventWindowTime values for ResponseOnEvent
[SWS_DM_01118]	Support of DEXT parameter DiagnosticResponseOnEventClass.responseOnEventSchedulerRate
[SWS_DM_01119]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumChangeOfDataIdentifierEvents
[SWS_DM_01120]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxNumComparisionOfValueEvents
[SWS_DM_01121]	Support of DEXT parameter DiagnosticResponseOnEventClass.maxSupportedDIDLength
[SWS_DM_01122]	Support of DEXT parameter DiagnosticResponseOnEvent- Class.maxNumberOfStoredDTCStatusChangedEvents
[SWS_DM_01123]	
[SWS_DM_01124]	
[SWS_DM_01125]	
[SWS_DM_01126]	
[SWS_DM_01127]	
[SWS_DM_01128]	
[SWS_DM_01129]	
[SWS_DM_01130]	
[SWS_DM_01131]	
[SWS_DM_01132]	
[SWS_DM_01133]	
[SWS_DM_01134]	
[SWS_DM_01136]	





Number	Heading
[SWS_DM_01137]	
[SWS_DM_01138]	
[SWS_DM_01139]	
[SWS_DM_01140]	
[SWS_DM_01141]	
[SWS_DM_01142]	
[SWS_DM_01143]	
[SWS_DM_01144]	
[SWS_DM_01145]	
[SWS_DM_01146]	
[SWS_DM_01147]	
[SWS_DM_01148]	
[SWS_DM_01149]	
[SWS_DM_01150]	
[SWS_DM_01151]	
[SWS_DM_01152]	
[SWS_DM_01153]	
[SWS_DM_01154]	
[SWS_DM_01155]	
[SWS_DM_01156]	
[SWS_DM_01157]	
[SWS_DM_01158]	
[SWS_DM_01159]	
[SWS_DM_01160]	
[SWS_DM_01161]	
[SWS_DM_01162]	
[SWS_DM_01163]	
[SWS_DM_01164]	
[SWS_DM_01165]	
[SWS_DM_01166]	
[SWS_DM_01167]	
[SWS_DM_01168]	
[SWS_DM_01169]	
[SWS_DM_01170]	
[SWS_DM_01171]	
[SWS_DM_01172]	
[SWS_DM_01173]	
[SWS_DM_01174]	
[SWS_DM_01175]	





Number	Heading
[SWS_DM_01176]	
[SWS_DM_01177]	
[SWS_DM_01178]	
[SWS_DM_01179]	
[SWS_DM_01180]	
[SWS_DM_01181]	
[SWS_DM_01182]	
[SWS_DM_01183]	
[SWS_DM_01184]	
[SWS_DM_01185]	
[SWS_DM_01186]	
[SWS_DM_01187]	
[SWS_DM_01188]	
[SWS_DM_01189]	
[SWS_DM_01190]	
[SWS_DM_01191]	
[SWS_DM_01192]	
[SWS_DM_01193]	
[SWS_DM_01194]	
[SWS_DM_01195]	
[SWS_DM_01196]	
[SWS_DM_01197]	
[SWS_DM_01198]	
[SWS_DM_01199]	
[SWS_DM_01200]	
[SWS_DM_01201]	
[SWS_DM_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





Number	Heading
[SWS_DM_01215]	Extend the DynamicAccessList
[SWS_DM_01216]	Revoke an authentication
[SWS_DM_01217]	Refresh timeouts
[SWS_DM_01218]	Building a new DynamicAccessList
[SWS_DM_01219]	Adding patterns to a DynamicAccessList
[SWS_DM_01220]	Adding wildcards to a DynamicAccessList
[SWS_DM_01221]	End patterns of a DynamicAccessList
[SWS_DM_01222]	Finalize a DynamicAccessList
[SWS_DM_01223]	Diagnostic service role verification
[SWS_DM_01224]	Diagnostic service dynamic access-rights verification
[SWS_DM_01225]	Response behavior of services without access rights
[SWS_DM_01226]	Support of UDS service authentication
[SWS_DM_01227]	Configuration of authentication types
[SWS_DM_01228]	Mandatory sub functions
[SWS_DM_01229]	Support for authentication per Diagnostic Conversation
[SWS_DM_01230]	Processing the verifyCertificateUnidirectional request
[SWS_DM_01231]	Handling Negative return values of <code>ara::diag::Authentication::VerifyCertificateUnidirectional</code> method
[SWS_DM_01232]	Handling unspecified negative return values of <code>ara::diag::Authentication::VerifyCertificateUnidirectional</code> method
[SWS_DM_01233]	Successful verification of verifyCertificateUnidirectional
[SWS_DM_01234]	Unexpected verifyCertificateUnidirectional from a different client
[SWS_DM_01235]	Processing the verifyCertificateBidirectional request
[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





Number	Heading
[SWS_DM_01246]	Processing the authenticationConfiguration request
[SWS_DM_01247]	Validation of the transmitCertificate certificateEvaluationId
[SWS_DM_01248]	Processing the transmitCertificate request
[SWS_DM_01249]	Handling Negative return values of ara::diag::Authentication::TransmitCertificate method
[SWS_DM_01250]	Handling unspecified negative return values of ara::diag::Authentication::TransmitCertificate method
[SWS_DM_01251]	Successful verification of transmitCertificate
[SWS_DM_- CONSTR_00961]	Limits of priority values

Table D.19: Added Traceables in R21-11

D.7.2 Changed Traceables in R21-11

Number	Heading
[SWS_DM_00018]	Internal debounce counter init
[SWS_DM_00022]	Internal debounce counter jump up behavior
[SWS_DM_00023]	Internal debounce counter jump down behavior
[SWS_DM_00039]	Resetting the internal debounce counter upon restarting an operation cycle
[SWS_DM_00040]	Definition of internal debounce counter reset
[SWS_DM_00086]	Resetting the internal debounce counter after clearing DTC
[SWS_DM_00088]	ControlDTCSetting influence (freeze)
[SWS_DM_00163]	Definition of a inhibited clear operation on single DTC
[SWS_DM_00164]	Definition of a inhibited clear operation for a group of DTCs
[SWS_DM_00230]	Check for supported subfunctions
[SWS_DM_00291]	
[SWS_DM_00293]	
[SWS_DM_00294]	
[SWS_DM_00296]	
[SWS_DM_00297]	
[SWS_DM_00298]	
[SWS_DM_00299]	
[SWS_DM_00300]	
[SWS_DM_00301]	
[SWS_DM_00302]	
[SWS_DM_00303]	
[SWS_DM_00304]	
[SWS_DM_00306]	





Number	Heading
[SWS_DM_00307]	
[SWS_DM_00309]	
[SWS_DM_00310]	
[SWS_DM_00311]	
[SWS_DM_00312]	
[SWS_DM_00313]	
[SWS_DM_00314]	
[SWS_DM_00315]	
[SWS_DM_00319]	
[SWS_DM_00322]	
[SWS_DM_00323]	
[SWS_DM_00325]	
[SWS_DM_00326]	
[SWS_DM_00327]	
[SWS_DM_00336]	
[SWS_DM_00337]	
[SWS_DM_00338]	
[SWS_DM_00378]	ControlDTCSetting influence (reset)
[SWS_DM_00384]	
[SWS_DM_00430]	Prioritization against all Diagnostic Conversations in active default session
[SWS_DM_00451]	
[SWS_DM_00452]	
[SWS_DM_00491]	Realisation of UDS service 0x86 ResponseOnEvent
[SWS_DM_00494]	Supported sub functions of ResponseOnEvent service
[SWS_DM_00512]	Data Type definitions for AUTOSAR Data Types in Implementation Types header files
[SWS_DM_00514]	
[SWS_DM_00515]	
[SWS_DM_00516]	
[SWS_DM_00517]	
[SWS_DM_00518]	
[SWS_DM_00519]	
[SWS_DM_00520]	
[SWS_DM_00521]	
[SWS_DM_00522]	
[SWS_DM_00523]	
[SWS_DM_00524]	
[SWS_DM_00525]	
[SWS_DM_00526]	





Number	Heading
[SWS_DM_00527]	
[SWS_DM_00528]	
[SWS_DM_00529]	
[SWS_DM_00530]	
[SWS_DM_00531]	
[SWS_DM_00532]	
[SWS_DM_00533]	
[SWS_DM_00534]	
[SWS_DM_00535]	
[SWS_DM_00536]	
[SWS_DM_00537]	
[SWS_DM_00538]	
[SWS_DM_00539]	
[SWS_DM_00540]	
[SWS_DM_00541]	
[SWS_DM_00542]	
[SWS_DM_00543]	
[SWS_DM_00544]	Use of general ara::diag errors
[SWS_DM_00545]	Definition Offer ara::diag errors
[SWS_DM_00546]	Definition Reporting ara::diag errors
[SWS_DM_00547]	Definition UDS NRC ara::diag errors
[SWS_DM_00548]	
[SWS_DM_00549]	
[SWS_DM_00550]	
[SWS_DM_00551]	
[SWS_DM_00552]	
[SWS_DM_00553]	
[SWS_DM_00554]	
[SWS_DM_00555]	
[SWS_DM_00556]	
[SWS_DM_00557]	
[SWS_DM_00558]	
[SWS_DM_00559]	
[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]	





Number	Heading
[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]	





Number	Heading
[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]	
[SWS_DM_00656]	
[SWS_DM_00657]	





Number	Heading
[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]	
[SWS_DM_00714]	
[SWS_DM_00715]	





Number	Heading
[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]	
[SWS_DM_00775]	
[SWS_DM_00776]	





Number	Heading
[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]	
[SWS_DM_00835]	
[SWS_DM_00836]	





Number	Heading
[SWS_DM_00837]	
[SWS_DM_00843]	Reset Service Instance fields on end of Diagnostic Conversation
[SWS_DM_00848]	Reading Diagnostic Data Identifier by typed DataIdentifier interface
[SWS_DM_00855]	Providing the VIN in DoIP protocol messages
[SWS_DM_00869]	UDS service TransferData (0x36) download processing
[SWS_DM_00871]	UDS service RequestTransferExit (0x37) download processing
[SWS_DM_00874]	Reporting kPrepassed or kPrefailed for events without an assigned debouncing algorithm
[SWS_DM_00875]	Internal debounce counter incrementation
[SWS_DM_00876]	Internal debounce counter decrementation
[SWS_DM_00879]	Application resetting the internal debounce counter
[SWS_DM_00882]	Enable condition influence on debouncing behavior (reset)
[SWS_DM_00902]	NumberOfStoredEntries
[SWS_DM_00906]	Writing Diagnostic Data Identifier by DataIdentifier interface
[SWS_DM_00918]	
[SWS_DM_00919]	
[SWS_DM_00935]	
[SWS_DM_00936]	
[SWS_DM_00937]	
[SWS_DM_00938]	
[SWS_DM_00939]	
[SWS_DM_00940]	Re-entrant ara::diag interface calls for service processing
[SWS_DM_00941]	Re-entrant ara::diag interface calls for DID read processing
[SWS_DM_00942]	Re-entrant ara::diag interface calls for DID write processing
[SWS_DM_00943]	Re-entrant ara::diag interface calls for DID read and write processing
[SWS_DM_00944]	Validity of re-entrant ara::diag interface calls for DID processing
[SWS_DM_00969]	Padding in case of failed data capturing
[SWS_DM_00971]	
[SWS_DM_00972]	
[SWS_DM_00973]	
[SWS_DM_00974]	
[SWS_DM_00975]	
[SWS_DM_00976]	
[SWS_DM_00977]	
[SWS_DM_00978]	
[SWS_DM_00979]	
[SWS_DM_00980]	
[SWS_DM_00989]	
[SWS_DM_00990]	





Number	Heading
[SWS_DM_00991]	
[SWS_DM_00992]	
[SWS_DM_00993]	
[SWS_DM_00994]	
[SWS_DM_00995]	
[SWS_DM_00996]	
[SWS_DM_00997]	
[SWS_DM_00998]	
[SWS_DM_00999]	
[SWS_DM_01000]	
[SWS_DM_01001]	
[SWS_DM_01002]	
[SWS_DM_01005]	
[SWS_DM_01006]	
[SWS_DM_01007]	
[SWS_DM_01009]	
[SWS_DM_01010]	
[SWS_DM_01011]	
[SWS_DM_01012]	
[SWS_DM_01013]	
[SWS_DM_01014]	
[SWS_DM_01016]	
[SWS_DM_01017]	
[SWS_DM_01023]	Calling ExecuteReset() if positive response shall be sent before reset
[SWS_DM_01027]	Event status bit transitions triggered by operation cycle restarting
[SWS_DM_01064]	
[SWS_DM_01065]	
[SWS_DM_01066]	
[SWS_DM_01067]	
[SWS_DM_01068]	
[SWS_DM_01069]	
[SWS_DM_09010]	
[SWS_DM_09011]	
[SWS_DM_09012]	
[SWS_DM_09013]	
[SWS_DM_09014]	
[SWS_DM_09015]	
[SWS_DM_09016]	
[SWS_DM_09017]	





Number	Heading
[SWS_DM_09018]	
[SWS_DM_09021]	

Table D.20: Changed Traceables in R21-11

D.7.3 Deleted Traceables in R21-11

Number	Heading
[SWS_DM_00004]	Operation cycle persistency
[SWS_DM_00028]	Debounce counter persistency
[SWS_DM_00165]	Considering only events referencing a DTC
[SWS_DM_00438]	Check Support of UDS service RequestUpload (0x35) in active session
[SWS_DM_00439]	Check Support of UDS service RequestUpload (0x35) on active security level
[SWS_DM_00440]	Check Support of UDS service TransferData (0x36) in active session
[SWS_DM_00441]	Check Support of UDS service TransferData (0x36) on active security level
[SWS_DM_00442]	Check Support of UDS service RequestTransferExit (0x37) in active session
[SWS_DM_00443]	Check Support of UDS service RequestTransferExit (0x37) on active security level
[SWS_DM_00446]	Check Support of UDS service RequestDownload (0x34) in active session
[SWS_DM_00447]	Check Support of UDS service RequestDownload (0x34) on active security level
[SWS_DM_00492]	Client Server communication
[SWS_DM_00495]	Start initialisation of ResponseOnEvent
[SWS_DM_00496]	Stop initialisation of ResponseOnEvent
[SWS_DM_00497]	Clear initialisation of ResponseOnEvent
[SWS_DM_00498]	Exclusive ResponseOnEvent ressources
[SWS_DM_00499]	Replacement of a not started ResponseOnEvent initialisation
[SWS_DM_00500]	Replacement of a started ResponseOnEvent initialisation
[SWS_DM_00501]	Behavior while trying ResponseOnEvent activation while ResponseOnEvent is not initialised
[SWS_DM_00564]	Monitor initialization for enable condition re-enabling reason
[SWS_DM_00565]	Monitor initialization for DTC setting re-enabling reason
[SWS_DM_00567]	Ignoring reported events for not started operation cycles
[SWS_DM_00750]	
[SWS_DM_00754]	
[SWS_DM_00756]	
[SWS_DM_00870]	UDS service TransferData (0x36) validation
[SWS_DM_00872]	UDS service RequestTransferExit (0x37) validation
[SWS_DM_00889]	Automatic starting of operation cycles





Number	Heading
[SWS_DM_00890]	Automatic ending of operation cycles
[SWS_DM_00891]	Restart of operation cycles
[SWS_DM_00892]	Operation cycles are only ended once
[SWS_DM_00893]	Triggering for snapshot record storage
[SWS_DM_00903]	Reading DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_00904]	Writing DiagnosticDataIdentifier configured for representing VIN
[SWS_DM_01008]	
[SWS_DM_01015]	

Table D.21: Deleted Traceables 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