Document Title	Acceptance Test Specification of Diagnostic Services
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	627
Document Classification	Standard

Document Status	Final
Part of AUTOSAR Release	1.0.0

Document Change History		
Release	**Changed by**	**Change Description**
1.0.0	AUTOSAR Release Management	Initial release, including test suites on • Dem DiagnosticMonitor • Dcm DataServices • Dcm RoutineServices

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
AT	Acceptance Test
CAN	Controller Area Network
ECU	Electronic Control Unit
ICC1	Implementation Conformance Class 1 (whole BSW & RTE)
ICC2	Implementation Conformance Class 2 (functional cluster of BSW)
ICC3	Implementation Conformance Class 3 (individual BSW modules)
IUT	Implementation under test
LT	Lower Tester
NM	Network Management
PCO	Point of Control and Observation
PDU	Protocol Data Unit
RfC	Request for Change
Rx	Reception
SUT	System Under Test
SWC	Software Component
TCP	Test Coordination Procedures
Tx	Transmission
UT	Upper Tester

2 Scope

The following test cases are used to verify the correct behavior of the diagnostic services.

Each test case documents for which releases of the AUTOSAR software specification it can be used:

- When test cases are known to be applicable for a release, this is mentioned in the “AUTOSAR Releases” field of the test case specifications.
  You can find a summary of the applicability of all test cases to the software specification releases in the “AUTOSAR_TR_ATSReleaseApplicability” document.
- When test cases are known to require adaptations (in their configuration requirements or test sequences), this is mentioned in the “Needed Adaptation to other Releases” field of the test case specifications.

3 General Test Objective and Approach

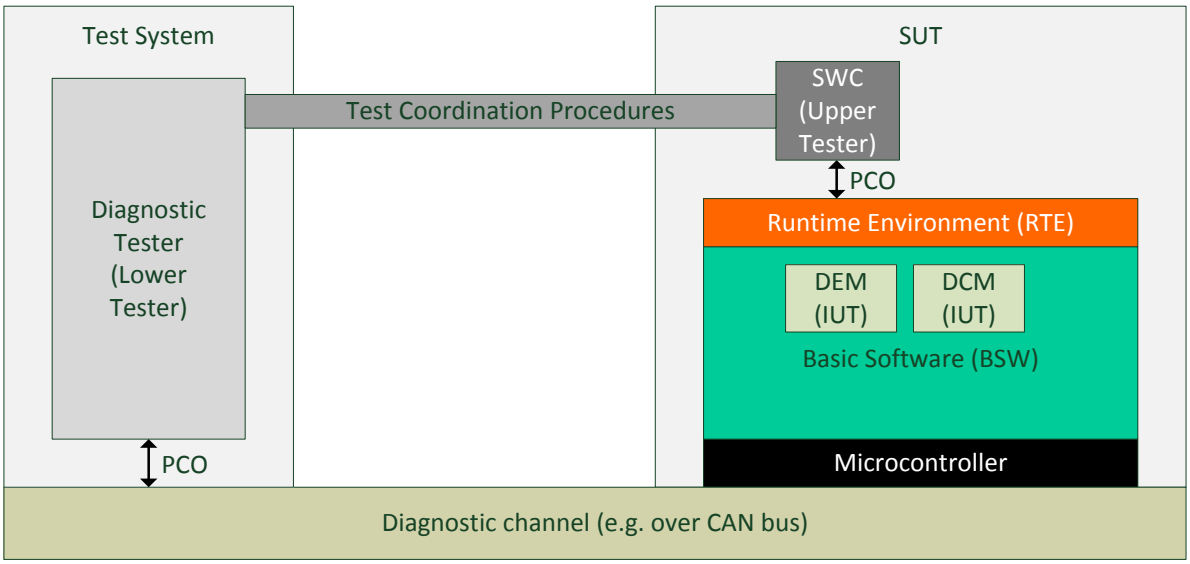
The diagnostics functionality of the AUTOSAR Basic Software implementation under test is tested by stimulating the SUT using diagnostic requests sent by the test environment (i.e. diagnostic tester) and by calling API functions that the DEM and DCM modules of the Basic Software provide to the SWCs (through the RTE).

The behavior of the diagnostic modules DEM and DCM is also observed on the diagnostic channel connecting the diagnostic tester with the SUT and at the interfaces of the SWCs which belong to the diagnostic API provided by the DEM and DCM modules.

3.1 Test System

3.1.1 Overview on Architecture

The basic test setup is depicted in the following figure:



The test cases require a SWC as Upper Tester and a diagnostic tester as Lower Tester.

3.1.2 Specific Requirements

The SWC can call diagnostic service functions and provides callback functions to be called by the DEM and DCM modules upon reception of specific diagnostic request. In this way, this SWC observes the diagnostic functionality under test.

The diagnostic tester sends UDS requests on the diagnostic channel and receives the UDS responses of the SUT. These responses are then evaluated by the test system with respect to the expectation from the test case.

3.1.3 Test Coordination Requirements

As observation of the SUT is done by the test cases at both the Lower Tester and the Upper Tester, a test coordination procedure for collecting the local test verdicts (at LT and UT) at one central place is required. It is up to the test system designer /

implementer to define that “central place” and to design and implement the test coordination functionality.

3.2 Test Configuration

A proper configuration of the diagnostics functionality in the DEM and DCM modules is a mandatory prerequisite for execution of the acceptance test cases specified in this document.

Although the objective of AUTOSAR Acceptance Tests is to use ICC1 level configuration files (i.e. System Description and SWC Description) for BSW configuration, the diagnostic test cases contain the configuration requirements on EcuC (ICC3) level.

This is due to the current practice of using EcuC parameters for configuring the diagnostic modules. Furthermore, a future diagnostic format on ICC1 level (planned for AUTOSAR R4.2.1) is expected to better fit the needs of diagnostic development than the current SWC ServiceNeeds on diagnostics. Therefore, the configuration requirements of the acceptance tests on diagnostic services shall later be transformed to this new format.

3.2.1 Required ECU Extract of System Description Files

An ECU Extract with the definitions for the diagnostic connection (e.g. using CAN TP) between SUT and Lower Tester (external diagnostic tester) is usually needed to configure the SUT. However, it is the user’s responsibility to create this ECU Extract.

3.2.2 Required ECU Configuration Description Files

The test cases require an ECU Configuration Value Description with definition of configuration parameters for the DEM and DCM modules. This description file (or alternatively the matching configuration with the configuration tooling) needs to be created by the user from the configuration requirements of the test cases.

3.2.3 Required Software Component Description Files

For the test cases on diagnostic functionality, the required definitions in the SWC Description (e.g. PortInterfaces) can be directly derived from the configuration requirements and interactions between the SWC and the RTE specified in the test cases. The SWC Description needs to be created by the user based on these requirements.

3.2.4 Mandatory vs. Customizable Parts

The configuration requirements on the “DCM Record Data Identifiers” as stated in the test cases are mandatory to be applied since they define specific DCM functionality that is in the focus of the tests.

The configuration parts in the ECU Configuration Value Description related to communication between ECU and diagnostic tester need to be customized by the test implementer according to user or system specific requirements.

3.3 Test Case Design

3.3.1 DEM related test cases

The test cases on the service interface “Dem_DiagnosticMonitor “ are based on use cases containing different situations of reporting an event by an application SWC to the DEM.

For each test case, a SWC is defined and needs to be implemented. This SWC executes the test steps defined in the test case which stimulate and observe the DEM through the SWC’s ports to the DEM services.

The functionality of retrieving events, getting the “fault detection counter”, getting the associated DTC and resetting the event is tested in each of the above test cases using re-usable test steps (see Ch. 4).

Enable and storage conditions are not tested here because the required functionality is located in a different service interface.

3.3.2 DCM related test cases

The test cases on the service interface “DataServices” (in R3.x: “Dcm_DidServices”) are based on the use cases made up by the related UDS services (ReadDataByIdentifier, WriteDataByIdentifier, ReadScalingDataByIdentifier, InputOutputControlByIdentifier).

However, the test cases focus on the interaction between the SWC and the DCM module (through the RTE). Therefore, DCM functionality that is internal and has no interference with SWCs (e.g. checking for validity of a request with respect to current diagnostic session or current security level) is not in scope of the tests.

The design approach for the test cases is to stimulate the SUT using UDS requests from the test environment and to observe the SUT’s behavior at the RTE interface of the DCM module and its UDS responses.

4 Re-usable Test Steps

Test steps that are re-used by multiple test cases are defined as “Test Step Groups”. They are defined together with the test cases in Chapter 5:

- Test Step Group: Reset event
- Test Step Group: Check event reset
- Test Step Group: Check event passed after failed
- Test Step Group: Check failed event

5 Test Cases

5.1 Test Cases on DiagnosticMonitor (Dem)

5.1.1 [ATS_DIAG_00032] Test Step Group: Reset event

Test Objective	Test Step Group: Reset event		
ID	ATS_DIAG_00032	**AUTOSAR Releases**	
Affected Modules		**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00331 DiagnosticEventManager: SWS_Dem_00187 DiagnosticEventManager: SWS_Dem_00051 DiagnosticEventManager: SWS_Dem_00333 DiagnosticEventManager: SWS_Dem_00052 DiagnosticEventManager: SWS_Dem_00053 DiagnosticEventManager: SWS_Dem_00036 DiagnosticEventManager: SWS_Dem_00379 DiagnosticEventManager: SWS_Dem_00204 DiagnosticEventManager: SWS_Dem_00185 DiagnosticEventManager: SWS_Dem_00195 DiagnosticEventManager: SWS_Dem_00196 DiagnosticEventManager: SWS_Dem_00197 DiagnosticEventManager: SWS_Dem_00203		
Requirements / Reference to Test Environment			
Configuration Parameters			
Summary	This test step group is used by the test cases.		
Needed Adaptation to other Releases			
Pre-conditions	1. Input parameter "eventID" has a value		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**LT:** Reset all DemEvents with associated DTC using UDS request "ClearDiagnosticInformation" (0x14FFFFFF)	SUT sends positive response (0x54)	
Step 2	**SWC:** Get status of eventID using Dem_GetEventStatus() [SWS_Dem_00195] [SWS_Dem_00051]	E_OK is returned	
Step 3	**SWC:** Check that the status bits are as expected [SWS_Dem_00036] [SWS_Dem_00379]	Bit 0 (TestFailed): 0 Bit 1 (TestFailedThisOperationCycle): 0	

		Bit 2 (PendingDTC): 0 Bit 3 (ConfirmedDTC): 0
Step 4	**SWC:** Get FAILED status of eventID using Dem_GetEventFailed() [SWS_Dem_00333] [SWS_Dem_00196] [SWS_Dem_00052]	E_OK is returned
Step 5	**SWC:** Check the FAILED status	FAILED status is FALSE [SWS_Dem_00187]
Step 6	**SWC:** Get TESTED status of eventID using Dem_GetEventTested() [SWS_Dem_00333] [SWS_Dem_00197] [SWS_Dem_00053]	E_OK is returned
Step 7	**SWC:** Check the TESTED status	TESTED status is FALSE
Step 8	**SWC:** Get fault detection counter of eventID using Dem_GetFaultDetectionCounter() [SWS_Dem_00203] [SWS_Dem_00204]	E_OK is returned
Step 9	**SWC:** Check the fault detection counter	Fault detection counter is 0 [SWS_Dem_00343]
Post-conditions		

5.1.2 [ATS_DIAG_00033] Test Step Group: Check event reset

Test Objective	Test Step Group: Check event reset		
ID	ATS_DIAG_00033	**AUTOSAR Releases**	
Affected Modules		**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00269 DiagnosticEventManager: SWS_Dem_00051 DiagnosticEventManager: SWS_Dem_00036 DiagnosticEventManager: SWS_Dem_00379 DiagnosticEventManager: SWS_Dem_00204 DiagnosticEventManager: SWS_Dem_00195 DiagnosticEventManager: SWS_Dem_00198 DiagnosticEventManager: SWS_Dem_00203		
Requirements / Reference to Test Environment			
Configuration Parameters			
Summary	This test step group is used by the test cases.		
Needed Adaptation to other Releases			

Pre-conditions	1. Input parameter "eventID" has a value	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC:** Get DTC associated with eventID using Dem_GetDTCOfEvent() [SWS_Dem_00198] [SWS_Dem_00269]	E_OK is returned
Step 2	**SWC:** Check the DTC value is equal to the configured one	DTC value is equal to the configured one
Step 3	**SWC:** Get status of eventID using Dem_GetEventStatus() [SWS_Dem_00195] [SWS_Dem_00051]	E_OK is returned
Step 4	**SWC:** Check that the status bits are as expected [SWS_Dem_00036] [SWS_Dem_00379]	Bit 0 (TestFailed): 0 Bit 1 (TestFailedThisOperationCycle): 0 Bit 2 (PendingDTC): 0 Bit 3 (ConfirmedDTC): 0 Bit 4 (TestNotCompletedSinceLastClear): don't care Bit 5 (TestFailedSinceLastClear): 0 Bit 6 (TestNotCompletedThisOperationCycle): don't care Bit 7 (WarningIndicatorRequested): 0
Step 5	**SWC:** Get fault detection counter of eventID using Dem_GetFaultDetectionCounter() [SWS_Dem_00203] [SWS_Dem_00204]	E_OK is returned
Step 6	**SWC:** Check the fault detection counter	Fault detection counter is 0 [SWS_Dem_00343]
Post-conditions		

5.1.3 [ATS_DIAG_00034] Test Step Group: Check event passed after failed

Test Objective	Test Step Group: Check event passed after failed		
ID	ATS_DIAG_00034	**AUTOSAR Releases**	
Affected Modules		**State**	reviewed
Trace to Requirement on Acceptance Test Document			

Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00269 DiagnosticEventManager: SWS_Dem_00051 DiagnosticEventManager: SWS_Dem_00333 DiagnosticEventManager: SWS_Dem_00052 DiagnosticEventManager: SWS_Dem_00053 DiagnosticEventManager: SWS_Dem_00036 DiagnosticEventManager: SWS_Dem_00379 DiagnosticEventManager: SWS_Dem_00204 DiagnosticEventManager: SWS_Dem_00195 DiagnosticEventManager: SWS_Dem_00196 DiagnosticEventManager: SWS_Dem_00197 DiagnosticEventManager: SWS_Dem_00198 DiagnosticEventManager: SWS_Dem_00203	
Requirements / Reference to Test Environment		
Configuration Parameters		
Summary	This test step group is used by the test cases.	
Needed Adaptation to other Releases		
Pre-conditions	1. Input parameter "eventID" has a value	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC:** Get DTC associated with eventID using Dem_GetDTCOfEvent() [SWS_Dem_00198] [SWS_Dem_00269]	E_OK is returned
Step 2	**SWC:** Check the DTC value is equal to the configured one	DTC value is equal to the configured one
Step 3	**SWC:** Get status of eventID using Dem_GetEventStatus() [SWS_Dem_00195] [SWS_Dem_00051]	E_OK is returned
Step 4	**SWC:** Check that the status bits are as expected [SWS_Dem_00036] [SWS_Dem_00379]	Bit 0 (TestFailed): 0 Bit 1 (TestFailedThisOperationCycle): 1 Bit 2 (PendingDTC): 1 Bit 3 (ConfirmedDTC): 1 Bit 4 (TestNotCompletedSinceLastClear): 1 Bit 5 (TestFailedSinceLastClear): 1 Bit 6 (TestNotCompletedThisOperationCycle): 0 Bit 7 (WarningIndicatorRequested): 0
Step 5	**SWC:** Get FAILED status of eventID using	E_OK is returned

	Dem_GetEventFailed() [SWS_Dem_00333] [SWS_Dem_00196] [SWS_Dem_00052]	
Step 6	**SWC:** Check the FAILED status	FAILED status is FALSE [SWS_Dem_00187]
Step 7	**SWC:** Get TESTED status of eventID using Dem_GetEventTested() [SWS_Dem_00333] [SWS_Dem_00197] [SWS_Dem_00053]	E_OK is returned
Step 8	**SWC:** Check the TESTED status	TESTED status is TRUE
Step 9	**SWC:** Get fault detection counter of eventID using Dem_GetFaultDetectionCounter() [SWS_Dem_00203] [SWS_Dem_00204]	E_OK is returned
Post-conditions		

5.1.4 [ATS_DIAG_00035] Test Step Group: Check failed event

Test Objective	Test Step Group: Check failed event		
ID	ATS_DIAG_00035	**AUTOSAR Releases**	
Affected Modules		**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00051 DiagnosticEventManager: SWS_Dem_00333 DiagnosticEventManager: SWS_Dem_00052 DiagnosticEventManager: SWS_Dem_00053 DiagnosticEventManager: SWS_Dem_00036 DiagnosticEventManager: SWS_Dem_00379 DiagnosticEventManager: SWS_Dem_00338 DiagnosticEventManager: SWS_Dem_00194 DiagnosticEventManager: SWS_Dem_00195 DiagnosticEventManager: SWS_Dem_00196 DiagnosticEventManager: SWS_Dem_00197		
Requirements / Reference to Test Environment			
Configuration Parameters			
Summary	This test step group is used by the test cases.		
Needed Adaptation to other Releases			
Pre-conditions	1. Input parameter "eventID" has a value		
Main Test Execution			
Test Steps			**Pass Criteria**

Step 1	**SWC:** Get status of eventID using Dem_GetEventStatus() [SWS_Dem_00195] [SWS_Dem_00051]	E_OK is returned
Step 2	**SWC:** Check that the status bits are as expected [SWS_Dem_00036] [SWS_Dem_00379]	Bit 0 (TestFailed): 1 Bit 1 (TestFailedThisOperationCycle): 1 Bit 2 (PendingDTC): 1 Bit 3 (ConfirmedDTC): 1 Bit 4 (TestNotCompletedSinceLastClear): 0 Bit 5 (TestFailedSinceLastClear): 1 Bit 6 (TestNotCompletedThisOperationCycle): 0 Bit 7 (WarningIndicatorRequested): 1: if indicator is configured for eventID 0: else
Step 3	**SWC:** Get FAILED status of eventID using Dem_GetEventFailed() [SWS_Dem_00333] [SWS_Dem_00196] [SWS_Dem_00052]	E_OK is returned
Step 4	**SWC:** Check the FAILED status	FAILED status is TRUE
Step 5	**SWC:** Get TESTED status of eventID using Dem_GetEventTested() [SWS_Dem_00333] [SWS_Dem_00197] [SWS_Dem_00053]	E_OK is returned
Step 6	**SWC:** Check the TESTED status	TESTED status is TRUE
Post-conditions		

5.1.5 [ATS_DIAG_00077] Reporting of an event without FreezeFrame and without debouncing

Test Objective	Reporting of an event without FreezeFrame and without debouncing		
ID	ATS_DIAG_00077	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00018		
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00183		
Requirements	SWC: A tester SWC that stimulates and observes the Dem API through the RTE.		

/ Reference to Test Environment		
Configuration Parameters	A Dem event E01 with DemDTC = 0x111111 must be configured with the following properties: - No associated FreezeFrame - No debouncing by Dem (i.e. definition of <DemDebounceMonitorInternal>) - DemAgingAllowed = FALSE - No DemEventFailureCycleCounterThreshold defined	
Summary	Ensure that a reported DEM event without FreezeFrame and without debouncing (i.e. immediately qualified) is stored correctly: Before reporting of E01, ensure that the DTC is not stored. Then, report event E01 and ensure that the DTC is stored.	
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [none]	
Pre-conditions	1. Event E01 is not FAILED 2. Operation cycle for event E01 has started	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC**: Execute “check event reset” group for event E01	All test steps passed
Step 2	**SWC**: Invoke Dem_SetEventStatus(E01, FAILED)	E_OK is returned
Step 3	**SWC**: Delay for 1 sec	
Step 4	**SWC**: Execute “check failed event” group for event E01	All test steps passed
Step 5	**SWC**: Execute “reset event” group for event E01	All test steps passed
Post-conditions	1. Event E01 is reset	

5.1.6 [ATS_DIAG_00078] Reporting of an event with counter-based debouncing and jump after event status change

Test Objective	Reporting of an event with counter-based debouncing and jump after event status change		
ID	ATS_DIAG_00078	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00018		

Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00413 DiagnosticEventManager: SWS_Dem_00526 DiagnosticEventManager: SWS_Dem_00414 DiagnosticEventManager: SWS_Dem_00415 DiagnosticEventManager: SWS_Dem_00416 DiagnosticEventManager: SWS_Dem_00417 DiagnosticEventManager: SWS_Dem_00418 DiagnosticEventManager: SWS_Dem_00419 DiagnosticEventManager: SWS_Dem_00422 DiagnosticEventManager: SWS_Dem_00423 DiagnosticEventManager: SWS_Dem_00424 DiagnosticEventManager: SWS_Dem_00425 DiagnosticEventManager: SWS_Dem_00183
Requirements / Reference to Test Environment	SWC: A tester SWC that stimulates and observes the Dem API through the RTE.
Configuration Parameters	A Dem event E02 must be configured with DemDTC = 0x222222 and the following properties: - DemDebounceCounterBased - DemDebounceCounterIncrementStepSize = 7 - DemDebounceCounterFailedThreshold = 127 - DemDebounceCounterDecrementStepSize = 17 - DemDebounceCounterPassedThreshold = -128 - DemDebounceCounterJumpUp = TRUE - DemDebounceCounterJumpUpValue = 0 - DemDebounceCounterJumpDown = TRUE - DemDebounceCounterJumpDownValue = 0 - DemAgingAllowed = FALSE - No DemEventFailureCycleCounterThreshold defined
Summary	Ensure that a reported DEM event is debounced (counter-based and with jump after event status change) and stored correctly: PREFAILED and PREPASSED events are reported repeatedly while the event status is checked. After the right number of PREPASS or PREFAILED reports, the event status is expected to change to FAILED or PASSED, respectively.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [none]
Pre-conditions	1. Event E02 is not FAILED 2. Operation cycle for event E02 has started

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC**: Execute “check event reset” group for event E02	All test steps passed
Step 2	**-- Debouncing of PREFAILED event --** **SWC**: *Execute the following LOOP for ceil(127 DIV DemDebounceCounterIncrementStepSize) - 1 = 18 times.*	

Step 3	**-- Start of LOOP with SWC actions --** **LOOP**: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 4	**LOOP**: Get FAILED status of E02 using Dem_GetEventFailed()	E_OK is returned
Step 5	**LOOP**: Check FAILED status	FAILED is FALSE
Step 6	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E02 to retrieve FDCounter	E_OK is returned
Step 7	**LOOP**: Check that FDCounter is (i+1) * DemDebounceCounterIncrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 8	**SWC**: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 9	**SWC**: Delay for 1 sec	
Step 10	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E02 to retrieve FDCounter	E_OK is returned
Step 11	**SWC**: Check that FDCounter is 127	FDCounter has the expected value
Step 12	**SWC**: Execute “check failed event” test steps for event E02	All test steps passed
Step 13	**-- Debouncing of PREPASSED event --** **SWC**: *Execute the following LOOP for ceil(128 DIV DemDebounceCounterDecrementStepSize) - 1 = 7 times*	
Step 14	**-- Start of LOOP with SWC actions --** LOOP: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREPASSED)	E_OK is returned
Step 15	**LOOP**: Get FAILED status of E02 using Dem_GetEventFailed()	E_OK is returned
Step 16	**LOOP**: Check FAILED status	FAILED is TRUE
Step 17	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E02 to retrieve FDCounter	E_OK is returned
Step 18	**LOOP**: Check that FDCounter is -(i+1) * DemDebounceCounterDecrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 19	**SWC**: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREPASSED)	E_OK is returned
Step 20	**SWC**: Delay for 1 sec	
Step 21	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E02 to	E_OK is returned

	retrieve FDCounter	
Step 22	**SWC**: Check that FDCounter is -128	FDCounter has the expected value
Step 23	**SWC**: Execute “check event passed after failed” test steps for event E02	All test steps passed
Step 24	**-- Debouncing of PREFAILED event --** **SWC**: *Execute the following LOOP for ceil(127 DIV DemDebounceCounterIncrementStepSize) - 1 = 18 times.*	
Step 25	**-- Sart of LOOP with SWC actions --** **LOOP**: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 26	**LOOP**: Get FAILED status of E02 using Dem_GetEventFailed()	E_OK is returned
Step 27	**LOOP**: Check FAILED status	FAILED is FALSE
Step 28	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E02 to retrieve FDCounter	E_OK is returned
Step 29	**LOOP**: Check that FDCounter is (i+1) * DemDebounceCounterIncrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 30	**SWC**: Invoke Dem_SetEventStatus(E02, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 31	**SWC**: Delay for 1 sec	
Step 32	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E02 to retrieve FDCounter	E_OK is returned
Step 33	**SWC**: Check that FDCounter is 127	FDCounter has the expected value
Step 34	**SWC**: Execute “check failed event” test steps for event E02	All test steps passed
Step 35	**SWC**: Execute “reset event” test steps for event E02	All test steps passed
Post-conditions	1. Event E02 is reset	

5.1.7 [ATS_DIAG_00085] Reporting of an event with counter-based debouncing and without jump

Test Objective	Reporting of an event with counter-based debouncing and without jump		
ID	ATS_DIAG_00085	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to	ATR: ATR_ATR_00018		

Requirement on Acceptance Test Document	
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00413 DiagnosticEventManager: SWS_Dem_00526 DiagnosticEventManager: SWS_Dem_00414 DiagnosticEventManager: SWS_Dem_00415 DiagnosticEventManager: SWS_Dem_00416 DiagnosticEventManager: SWS_Dem_00417 DiagnosticEventManager: SWS_Dem_00418 DiagnosticEventManager: SWS_Dem_00419 DiagnosticEventManager: SWS_Dem_00422 DiagnosticEventManager: SWS_Dem_00424 DiagnosticEventManager: SWS_Dem_00183
Requirements / Reference to Test Environment	SWC: A tester SWC that stimulates and observes the Dem API through the RTE
Configuration Parameters	A Dem event E03 with DemDTC = 0x333333 must be configured with the following properties: - DemDebounceCounterBased - DemDebounceCounterIncrementStepSize = 7 - DemDebounceCounterFailedThreshold = 127 - DemDebounceCounterDecrementStepSize = 17 - DemDebounceCounterPassedThreshold = -128 - DemDebounceCounterJumpUp = FALSE - DemDebounceCounterJumpUpValue = 0 - DemDebounceCounterJumpDown = FALSE - DemDebounceCounterJumpDownValue = 0 - DemAgingAllowed = FALSE - No DemEventFailureCycleCounterThreshold defined
Summary	Ensure that a reported DEM event is debounced (counter-based without jump) and stored correctly: PREFAILED and PREPASSED events are reported repeatedly while the event status is checked. After the right number of PREPASS or PREFAILED reports, the event status is expected to change to FAILED or PASSED, respectively.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [none]
Pre-conditions	1. Event E03 is not FAILED 2. Operation cycle for event E03 has started

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC**: Execute “check event reset” group for event E03	All test steps passed
Step 2	**-- Debouncing of PREFAILED event --** **SWC**: *Execute the following LOOP for ceil(127 DIV*	

	DemDebounceCounterIncrementStepSize) - 1 = 18 times.	
Step 3	**-- Start of LOOP with SWC actions --** **LOOP**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 4	**LOOP**: Get FAILED status of E03 using Dem_GetEventFailed()	E_OK is returned
Step 5	**LOOP**: Check FAILED status	FAILED is FALSE
Step 6	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 7	**LOOP**: Check that FDCounter is (i+1) * DemDebounceCounterIncrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 8	**SWC**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 9	**SWC**: Delay for 1 sec	
Step 10	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 11	**SWC**: Check that FDCounter is 127	FDCounter has the expected value
Step 12	**SWC**: Execute “check failed event” test steps for event E03	All test steps passed
Step 13	**-- Debouncing of PREPASSED event --** **SWC**: *Execute the following LOOP for ceil(255 DIV DemDebounceCounterDecrementStepSize) - 1 = 14 times*	
Step 14	**-- Start of LOOP with SWC actions --** **LOOP**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREPASSED)	E_OK is returned
Step 15	**LOOP**: Get FAILED status of E03 using Dem_GetEventFailed()	E_OK is returned
Step 16	**LOOP**: Check FAILED status	FAILED is TRUE
Step 17	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 18	**LOOP**: Check that FDCounter is 127 - (i+1) * DemDebounceCounterDecrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 19	**SWC**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREPASSED)	E_OK is returned

Step 20	**SWC**: Delay for 1 sec	
Step 21	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 22	**SWC**: Check that FDCounter is -128	FDCounter has the expected value
Step 23	**SWC**: Execute “check event passed after failed” test steps for event E03	All test steps passed
Step 24	**-- Debouncing of PREFAILED event --** **SWC**: *Execute the following LOOP for ceil(255 DIV DemDebounceCounterIncrementStepSize) - 1 = 36 times*	
Step 25	**-- Start of LOOP with SWC actions --** **LOOP**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 26	**LOOP**: Get FAILED status of E03 using Dem_GetEventFailed()	E_OK is returned
Step 27	**LOOP**: Check FAILED status	FAILED is FALSE
Step 28	**LOOP**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 29	**LOOP**: Check that FDCounter is -128 + (i+1) * DemDebounceCounterIncrementStepSize (with i being the number of repetitions counting from 0) **-- End of LOOP with SWC actions --**	FDCounter has the expected value
Step 30	**SWC**: Invoke Dem_SetEventStatus(E03, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned
Step 31	SWC: Delay for 1 sec	
Step 32	**SWC**: Invoke Dem_GetFaultDetectionCounter() on E03 to retrieve FDCounter	E_OK is returned
Step 33	**SWC**: Check that FDCounter is 127	FDCounter has the expected value
Step 34	**SWC**: Execute “check failed event” test steps for event E03	All test steps passed
Step 35	**SWC**: Execute “reset event” test steps for event E03	All test steps passed
Post-conditions	1. Event E03 is reset	

5.1.8 [ATS_DIAG_00245] Reporting of an event with time-based debouncing

Test Objective	Reporting of an event with time-based debouncing		
ID	ATS_DIAG_00245	**AUTOSAR Releases**	4.0.3|4.1.1

Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00018		
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00413 DiagnosticEventManager: SWS_Dem_00527 DiagnosticEventManager: SWS_Dem_00426 DiagnosticEventManager: SWS_Dem_00427 DiagnosticEventManager: SWS_Dem_00428 DiagnosticEventManager: SWS_Dem_00430 DiagnosticEventManager: SWS_Dem_00432 DiagnosticEventManager: SWS_Dem_00434 DiagnosticEventManager: SWS_Dem_00183		
Requirements / Reference to Test Environment	SWC: A tester SWC that stimulates and observes the Dem API through the RTE.		
Configuration Parameters	A Dem event E04 with DemDTC = 0x444444 must be configured with the following properties: - DemDebounceTimeBase - DemDebounceTimeFailedThreshold = 6000 milliseconds - DemDebounceTimePassedThreshold = 4000 milliseconds - DemAgingAllowed = FALSE - No DemEventFailureCycleCounterThreshold defined		
Summary	Report PREFAILED event and ensure that the event is not FAILED until 6 sec (DemTimeFailedThreshold) later and then set to FAILED. Afterwards, report PREPASSED event and ensure that the event is still FAILED until 4 sec (DemTimePassedThreshold) later and then set to PASSED.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [none]		
Pre-conditions	1. Event E04 is not FAILED. 2. Operation cycle for event E04 has started.		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**SWC**: Execute “check event reset” group for event E04	All test steps passed	
Step 2	**-- Debouncing of PREFAILED event --** **SWC**: Invoke Dem_SetEventStatus(E04, DEM_EVENT_STATUS_PREFAILED)	E_OK is returned	
Step 3	**SWC**: Get FAILED status of E04 using Dem_GetEventFailed() for multiple times during the next DemTimeFailedThreshold = 6 seconds and check it	FAILED status is always FALSE	
Step 4	**SWC**: Get FAILED status of E04 using	E_OK is returned	

	Dem_GetEventFailed()	
Step 5	**SWC**: Check FAILED status	FAILED status is TRUE
Step 6	**SWC**: Execute “check failed event” test steps for event E04	All test steps passed
Step 7	**-- Debouncing of PREPASSED event --** **SWC**: Invoke Dem_SetEventStatus(E04, DEM_EVENT_STATUS_PREPASSED)	E_OK is returned
Step 8	**SWC**: Invoke Dem_GetEventFailed() on E04	E_OK is returned
Step 9	**SWC**: Get FAILED status of E04 using Dem_GetEventFailed() for multiple times during the next DemTimePassedThreshold = 4 seconds and check it	FAILED status is always TRUE
Step 10	**SWC**: Get FAILED status of E04 using Dem_GetEventFailed()	E_OK is returned
Step 11	**SWC**: Check FAILED status	FAILED status is FALSE
Step 12	**SWC**: Execute “check event passed after failed” test steps for event E04	All test steps passed
Step 13	**SWC**: Execute “reset event” test steps for event E04	All test steps passed
Post-conditions	1. Event E04 is reset.	

5.1.9 [ATS_DIAG_00246] Reporting of an event with FreezeFrame and without pre-storage

Test Objective	Reporting of an event with FreezeFrame and without pre-storage		
ID	ATS_DIAG_00246	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00018		
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00337 DiagnosticEventManager: SWS_Dem_00183		
Requirements / Reference to Test Environment	SWC: A tester SWC that stimulates and observes the Dem API through the RTE.		
Configuration Parameters	A Dem event E05 with DemDTC = 0x555555 must be configured with the following properties: - DemDebounceMonitorInternal - FreezeFrame contains variable FreezeFrameCounter (uint32) - DemFreezeFrameCapture = DEM_TRIGGER_TESTFAILED - DemAgingAllowed = FALSE		

	- No DemEventFailureCycleCounterThreshold defined The SWC provides the FreezeFrame data through a DID port to the DEM. The SWC's variable FreezeFrameCounter is incremented by 1 (with wrap-around at maximum value) during each periodic invocation of the SWC's main runnable (periodicity << 1 sec). The initial value of the FreezeFrameCounter is arbitrary.
Summary	Report a failed event and memorize the current FreezeFrameCounter value. Then, check that the event is failed and that the associated FreezeFrame contains the FreezeFrameCounter with the memorized value.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [low] In R3.2, the API function Dem_GetEventFreezeFrameData() does not exist. So the test steps related to this API need to be removed (only retrieving of FreezeFrames by diagnostic tester is used).
Pre-conditions	1. Event E05 is not FAILED. 2. Operation cycle for event E05 has started.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC**: Execute "check event reset" group for event E05	All test steps passed
Step 2	**SWC**: Memorize the current value of FreezeFrameCounter in FFC	
Step 3	**SWC**: Invoke Dem_SetEventStatus(E05, FAILED)	E_OK is returned
Step 4	**SWC**: Delay for 1 sec	
Step 5	**SWC**: Execute "check failed event" test steps for event E05	All test steps passed
Step 6	**DT**: Read out DemDTC = 0x555555 and store the associated FreezeFrame into FF1	Retrieving FreezeFrame successful
Step 7	**SWC**: Check the FreezeFrameCounter of the FreezeFrame FF1	FreezeFrameCounter value is the same as FFC
Step 8	**SWC**: Invoke Dem_GetEventFreezeFrameData() to retrieve FreezeFrame of DemDTC=0x555555 and store it into FF2	Retrieving FreezeFrame successful
Step 9	**SWC**: Check the FreezeFrameCounter of the FreezeFrame FF2	FreezeFrameCounter value is the same as FFC
Step 10	**SWC**: Execute "reset event" test steps for event E05	All test steps passed
Post-conditions	Event E05 is reset.	

5.1.10 [ATS_DIAG_00247] Reporting of an event with pre-stored FreezeFrame

Test Objective	Reporting of an event with pre-stored FreezeFrame		
ID	ATS_DIAG_00247	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00018		
Trace to R4.1.1 Item	DiagnosticEventManager: SWS_Dem_00330 DiagnosticEventManager: SWS_Dem_00039 DiagnosticEventManager: SWS_Dem_00040 DiagnosticEventManager: SWS_Dem_00461 DiagnosticEventManager: SWS_Dem_00261 DiagnosticEventManager: SWS_Dem_00002 DiagnosticEventManager: SWS_Dem_00189 DiagnosticEventManager: SWS_Dem_00464 DiagnosticEventManager: SWS_Dem_00191 DiagnosticEventManager: SWS_Dem_00478 DiagnosticEventManager: SWS_Dem_00479 DiagnosticEventManager: SWS_Dem_00183 DiagnosticEventManager: SWS_Dem_00558		
Requirements / Reference to Test Environment	SWC: A tester SWC that stimulates and observes the Dem API through the RTE.		
Configuration Parameters	A Dem event E06 with DemDTC = 0x666666 must be configured with the following properties: - DemDebounceMonitorInternal - FreezeFrame contains variable FreezeFrameCounter (uint32) - DemFreezeFrameCapture = DEM_TRIGGER_TESTFAILED - DemAgingAllowed = FALSE - No DemEventFailureCycleCounterThreshold defined The SWC provides the FreezeFrame data through a DID port to the DEM. The SWC's variable FreezeFrameCounter is incremented by 1 (with wrap-around at maximum value)during each periodic invocation of the SWC's main runnable (periodicity << 1 sec). The initial value of the FreezeFrameCounter is arbitrary.		
Summary	Memorize the current FreezeFrameCounter value and pre-store a FreezeFrame. After 2 seconds, report a failed event. Then, check that the event is failed and that the associated FreezeFrame contains the FreezeFrameCounter with the memorized value.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed). Test Steps: [low] In R3.2, the API function Dem_GetEventFreezeFrameData() does not exist. So the test steps related to this API need to be removed (only retrieving of FreezeFrames by diagnostic tester is used).		

Pre-conditions	1. Event E06 is not FAILED. 2. Operation cycle for event E07 has started.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC**: Execute “check event reset” group for event E06	All test steps passed
Step 2	**SWC**: Memorize the current value of FreezeFrameCounter in FFC	
Step 3	**SWC**: Invoke Dem_PrestoreFreezeFrame(E06)	E_OK is returned
Step 4	**SWC**: Delay for 2 seconds	
Step 5	**SWC**: Invoke Dem_SetEventStatus(E06, FAILED)	E_OK is returned
Step 6	**SWC**: Delay for 1 second	
Step 7	**SWC**: Execute “check failed event” test steps for event E06	All test steps passed
Step 8	**DT**: Read out DemDTC = 0x666666 and the associated FreezeFrame into FF1	Retrieving FreezeFrame FF1 successful
Step 9	**SWC**: Check the FreezeFrameCounter of FreezeFrame FF1	FreezeFrameCounter has the same value as FFC
Step 10	**SWC**: Invoke Dem_GetEventFreezeFrameData() to retrieve FreezeFrame of DemDTC=0x666666 into FF2	Retrieving FreezeFrame FF2 successful
Step 11	**SWC**: Check the FreezeFrameCounter of FreezeFrame FF2	FreezeFrameCounter has the same value as FFC
Step 12	**SWC**: Execute “reset event” test steps for event E06	All test steps passed
Post-conditions	1. Event E06 is reset.	

5.1.11 [ATS_DIAG_00248] Reporting of an event after pre-storing and clearing FreezeFrame

Test Objective	Reporting of an event after pre-storing and clearing FreezeFrame		
ID	ATS_DIAG_00248	**AUTOSAR Releases**	4.1.1
Affected Modules	Dem	**State**	reviewed
Trace to Requirement on Acceptance	ATR: ATR_ATR_00018		

<table>
<tr><td>Test Document</td><td></td></tr>
<tr><td>Trace to R4.1.1 Item</td><td>DiagnosticEventManager: SWS_Dem_00330
DiagnosticEventManager: SWS_Dem_00039
DiagnosticEventManager: SWS_Dem_00040
DiagnosticEventManager: SWS_Dem_00461
DiagnosticEventManager: SWS_Dem_00261
DiagnosticEventManager: SWS_Dem_00002
DiagnosticEventManager: SWS_Dem_00189
DiagnosticEventManager: SWS_Dem_00464
DiagnosticEventManager: SWS_Dem_00050
DiagnosticEventManager: SWS_Dem_00478
DiagnosticEventManager: SWS_Dem_00479
DiagnosticEventManager: SWS_Dem_00183
DiagnosticEventManager: SWS_Dem_00558</td></tr>
<tr><td>Requirements / Reference to Test Environment</td><td>SWC: A tester SWC that stimulates and observes the Dem API through the RTE. It has to provide the FreezeFrameCounter through a DID port to the DEM. The SWC has to increment the FreezeFrameCounter by 1 (with wrap-around at maximum value) during each periodic invocation of the SWC’s cyclic runnable (periodicity << 1 sec). The initial value of the FreezeFrameCounter is arbitrary.</td></tr>
<tr><td>Configuration Parameters</td><td>A Dem event E07 with DemDTC = 0x777777 must be configured with the following properties:
- DemDebounceMonitorInternal
- FreezeFrame contains variable FreezeFrameCounter (uint32)
- DemFreezeFrameCapture = DEM_TRIGGER_TESTFAILED
- DemAgingAllowed = FALSE
- No DemEventFailureCycleCounterThreshold defined

The SWC provides the FreezeFrame data through a DID port to the DEM. The SWC’s variable FreezeFrameCounter is incremented by 1 during each invocation of the SWC’s main runnable.</td></tr>
<tr><td>Summary</td><td>Pre-store a FreezeFrame. After 2 seconds, clear the pre-stored FreezeFrame, report a failed event and memorize the current FreezeFrameCounter value.

Then, check that the event is failed and that the associated FreezeFrame contains the FreezeFrameCounter with the memorized value.</td></tr>
<tr><td>Needed Adaptation to other Releases</td><td>Needed Adaptation for Release 3.2.2
<table><tr><td>Configuration: [low]

Test Steps: [low]</td><td>Some configuration parameters have different names in R3.2 than in R4.x (e.g.. DemHealingAllowed instead of DemAgingAllowed).

In R3.2, the API function Dem_GetEventFreezeFrameData() does not exist. So the test steps related to this API need to be removed (only retrieving of FreezeFrames by diagnostic tester is used).</td></tr></table></td></tr>
<tr><td>Pre-conditions</td><td>1. Event E07 is not FAILED.
2. Operation cycle for event E07 has started.</td></tr>
<tr><td colspan="2">Main Test Execution</td></tr>
<tr><td colspan="2">Test Steps</td><td>Pass Criteria</td></tr>
<tr><td>Step 1</td><td>SWC: Execute “check event reset” group for event E07</td><td>All test steps passed</td></tr>
</table>

Step 2	**SWC**: Invoke Dem_PrestoreFreezeFrame(E07)	E_OK is returned
Step 3	**SWC**: Delay for 2 seconds	
Step 4	**SWC**: Invoke Dem_ClearPrestoredFreezeFrame(E07)	E_OK is returned
Step 5	**SWC**: Memorize the current value of FreezeFrameCounter in FFC	
Step 6	**SWC**: Invoke Dem_SetEventStatus(E07, FAILED)	E_OK is returned
Step 7	**SWC**: Delay for 1 second	
Step 8	**SWC**: Execute “check failed event” test steps for event E07	All test steps passed
Step 9	*Alternatively:* **DT**: Read out DemDTC = 0x777777 and the associated FreezeFrame. *Or:* **SWC**: Invoke Dem_GetEventFreezeFrameData() to retrieve FreezeFrame of DemDTC=0x777777	Retrieving FreezeFrame successful
Step 10	**SWC**: Check the FreezeFrameCounter of the above FreezeFrame	FreezeFrameCounter has the same value as FFC
Step 11	**SWC**: Execute “reset event” test steps for event E07	All test steps passed
Post-conditions	1. Event E07 is reset.	

5.2 Test Cases on DataServices (Dcm)

5.2.1 [ATS_DIAG_00022] Writing and reading of data with fixed length

Test Objective	Writing and reading of data with fixed length		
ID	ATS_DIAG_00022	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00253 DiagnosticCommunicationManager: SWS_Dcm_00439 DiagnosticCommunicationManager: SWS_Dcm_00437 DiagnosticCommunicationManager: SWS_Dcm_00255 DiagnosticCommunicationManager: SWS_Dcm_00395		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	A data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE11 - DcmDspDidInfo.DcmDspDidDynamicallyDefined = FALSE - DcmDspDataSize = 32 - DcmDspDataInfo.DcmDspDataFixedLength = TRUE - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - Callbacks for “ConditionCheckRead”, “ReadData” and “WriteData” mapped to runnables of the SWC		
Summary	Verify that writing and reading of DID data with fixed length works correctly: First, the DID data is read. This value is then changed and written back to the DID. When again reading the DID data, the changed value is expected.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Test Steps: [low] Some parameter names for DID configuration are different in R3.2.2. In R3.2.2, the callout "ConditionCheckWrite" is available (but not in R4.x) and should therefore be checked additionally after LT has sent the "WriteDataByIdentifier" request.		
Pre-conditions	1. UDS connection between LT and SUT established 2. SWC: The variable “val0” is initialized with a random value of type UINT32		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**LT:** Send UDS request “ReadDataByIdentifier” for DID=0xFE11 to SUT		
Step 2	**SWC:** In runnable configured as “ConditionCheckRead” callback, return 0	Runnable configured as “ConditionCheckRead” callback	

	as error code and E_OK as return value	is called by the RTE [SWS_Dcm_00439]
Step 3		
Step 4	**SWC:** In runnable configured as "ReadData" callback, return the value "val0" as DID data and E_OK as return value	Runnable configured as "ReadData" callback is called by the RTE [SWS_Dcm_00437]
Step 5	**LT:** Receive UDS response from SUT and store received DID value in "val1"	Positive response code received
Step 6	**LT:** Calculate "val1" = "val1" XOR 0x01234567	
Step 7	**LT:** Send UDS request "WriteDataByIdentifier" for DID=0xFE11 and with value "val1" to SUT	
Step 8	**SWC:** In runnable configured as "WriteData" callback, store the received DID value to "val0" , return 0 as error code and E_OK as return value	Runnable configured as "WriteData" callback is called by the RTE [SWS_Dcm_00395]
Step 9	**LT:** Receive UDS response from SUT	Positive response code is received
Step 10	**LT:** Send UDS request "ReadDataByIdentifier" for DID=0xFE11 to SUT	
Step 11	**SWC:** In runnable configured as "ConditionCheckRead" callback, return E_OK as return value	Runnable configured as "ConditionCheckRead" callback is called by the RTE [Dcm439]
Step 12	**SWC:** In runnable configured as "ReadData" callback, return the value "val0" as DID data and E_OK as return value	Runnable configured as "ReadData" callback is called by the RTE [SWS_Dcm_00437]
Step 13	**LT:** Receive UDS response from SUT and store received DID value in "val2"	Positive response code is received
Step 14	**LT:** Compare "val1" and "val2"	"val1" and "val2" have the same value
Post-conditions	None	

5.2.2 [ATS_DIAG_00023] Writing and reading of data with dynamic length

Test Objective	Writing and reading of data with dynamic length		
ID	ATS_DIAG_00023	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00253 DiagnosticCommunicationManager: SWS_Dcm_00439		

	DiagnosticCommunicationManager: SWS_Dcm_00436 DiagnosticCommunicationManager: SWS_Dcm_00437 DiagnosticCommunicationManager: SWS_Dcm_00255 DiagnosticCommunicationManager: SWS_Dcm_00395
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT
Configuration Parameters	A data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE22 - DcmDspDidInfo.DcmDspDidDynamicallyDefined = FALSE - DcmDspDataSize = 256 - DcmDspDataInfo.DcmDspDataFixedLength = FALSE - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - Callbacks for “ConditionCheckRead”, “ReadDataLength”, “ReadData” and “WriteData” mapped to runnables of the SWC
Summary	Verify that writing and reading of DID data with dynamic length works correctly: Execute the following for the data lengths 1, 5, 16, 23 and 32: First, the DID data is read. This value is then changed and written back to the DID. When again reading the DID data, the changed value is expected.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Test Steps: [low] Some parameter names for DID configuration are different in R3.2.2. In R3.2.2, the callout "ConditionCheckWrite" is available (but not in R4.x) and should therefore be checked additionally after LT has sent the "WriteDataByIdentifier" request.
Pre-conditions	1. UDS connection between LT and SUT established
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	Execute the following loop multiple times with variable “len” = 1, 5, 16, 23, 32 for each iteration.
Step 2	**LOOP: LT:** Send UDS request “ReadDataByIdentifier” for DID=0xFE22 to SUT
Step 3	**LOOP: SWC:** Initialize the variable “val0” with random data of “len” bytes length
Step 4	**LOOP: SWC:** In runnable configured as “ConditionCheckRead” callback, return 0 as error code and E_OK as return value Runnable configured as “ConditionCheckRead” callback is called by the RTE [SWS_Dcm_00439]
Step 5	**LOOP: SWC:** In runnable configured as “ReadDataLength” callback, return the current data length “len” and E_OK as return value Runnable configured as “ReadDataLength” callback is called by the RTE [SWS_Dcm_00436]
Step 6	**LOOP: SWC:** In runnable configured as “ReadData” callback, return the value “val0” as DID data and E_OK as return value Runnable configured as “ReadData” callback is called by the RTE [SWS_Dcm_00437]

Step 7	**LOOP: LT:** Receive UDS response from SUT and store received DID value in “val1”	Positive response code received
Step 8	**LOOP: LT:** “val1” = “val1” XOR 0x0123456789ABCDEF0102… (“len” bytes length)	
Step 9	**LOOP: LT:** Send UDS request “WriteDataByIdentifier” for DID=0xFE22 and with value “val1” to SUT	
Step 10	**LOOP: SWC:** In runnable configured as “WriteData” callback, store the received DID value to “val0”, return 0 as error code and E_OK as return value	Runnable configured as “WriteData” callback is called by the RTE [SWS_Dcm_00395]
Step 11	**LOOP: LT:** Receive UDS response from SUT	Positive response code is received
Step 12	**LOOP: LT:** Send UDS request “ReadDataByIdentifier” for DID=0xFE22 to SUT	
Step 13	**LOOP: SWC:** In runnable configured as “ConditionCheckRead” callback, return 0 as error code and E_OK as return value	Runnable configured as “ConditionCheckRead” callback is called by the RTE [SWS_Dcm_00439]
Step 14	**LOOP: SWC:** In runnable configured as “ReadData” callback, return the value “val0” as DID data and E_OK as return value	Runnable configured as “ReadData” callback is called by the RTE [SWS_Dcm_00437]
Step 15	**LOOP: LT:** Receive UDS response from SUT and store received DID value in “val2”	Positive response code is received
Step 16	**LOOP: LT:** Compare “val1” and “val2”	“val1” and “val2” have the same length and value
Post-conditions	None	

5.2.3 [ATS_DIAG_00024] Reading data rejected by SWC

Test Objective	Reading data rejected by SWC		
ID	ATS_DIAG_00024	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00253 DiagnosticCommunicationManager: SWS_Dcm_00439		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT. In the SWC’s runnable configured as “ReadData” callback, the counter “cnt_read” counts the number of invocations of that runnable. LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		

Configuration Parameters	The data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE33 - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - Callbacks for “ConditionCheckRead” and “ReadData” mapped to runnables of the SWC - Further DID configuration parameters are arbitrary.
Summary	Verify that if SWC rejects reading the data is indeed not read: When the read request for a DID is passed to the SWC and the SWC rejects it, the SUT must not call the read callback but send a negative response code.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some parameter names for DID configuration are different in R3.2.2. Test Steps: [none]
Pre-conditions	1. UDS connection between LT and SUT established 2. In the runnable configured as “ReadData” callback, the counter “cnt_read” that counts the number of invocations of that runnable is initialized with 0.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**LT:** Send UDS request “ReadDataByIdentifier” for DID=0xFE33 to SUT	
Step 2	**SWC:** In runnable configured as “ConditionCheckRead” callback, return 0x22 (conditionsNotCorrect) as error code and E_NOT_OK as return value	Runnable configured as “ConditionCheckRead” callback is called by the RTE [SWS_Dcm_00439]
Step 3	**LT:** Receive UDS response from SUT with negative response code	Negative response code = 0x22
Step 4	**SWC:** Check that the runnable configured as “ReadData” callback has not been called.	Counter “cnt_read” of runnable for “ReadData” callback is still 0
Post-conditions	None	

5.2.4 [ATS_DIAG_00025] Writing data rejected by SWC

Test Objective	Writing data rejected by SWC		
ID	ATS_DIAG_00025	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00255 DiagnosticCommunicationManager: SWS_Dcm_00395		
Requirements	SWC: A tester SWC that interacts with the RTE interface of the SUT.		

/ Reference to Test Environment	LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT
Configuration Parameters	The data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE33 (re-used from AT_DCM_03) - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - Callback for “WriteData” mapped to runnables of the SWC - Further DID configuration parameters are arbitrary.
Summary	Verify that if SWC rejects writing the data, the SUT responds with the appropriate negative response code: When the write request for a DID is passed to the SWC and the SWC rejects it, the SUT must respond with a negative response code.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Test Steps: [middle] Some parameter names for DID configuration are different in R3.2.2. In R3.2.2, a specific callout "ConditionCheckWrite" exists which is called by the DCM before the callout of "WriteData". So the SWC has to reject the request using "ConditionCheckWrite" and the test case has to check that "WriteData" must not be called by the SUT.
Pre-conditions	1. UDS connection between LT and SUT established

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**LT:** Send UDS request “WriteDataByIdentifier” for DID=0xFE33 to SUT	
Step 2	**SWC:** In runnable configured as “WriteData” callback, return 0x22 (conditionsNotCorrect) as error code and E_NOT_OK as return value	Runnable configured as “WriteData” callback is called by the RTE [SWS_Dcm_00395]
Step 3	**LT:** Receive UDS response from SUT with negative response code	Negative response code = 0x22
Post-conditions	None	

5.2.5 [ATS_DIAG_00026] Writing and reading of data with endianness conversion

Test Objective	Writing and reading of data with endianness conversion		
ID	ATS_DIAG_00026	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance	ATR: ATR_ATR_00019		

Test Document		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00253 DiagnosticCommunicationManager: SWS_Dcm_00638 DiagnosticCommunicationManager: SWS_Dcm_00255	
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT	
Configuration Parameters	A data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE55 - DcmDspDidInfo.DcmDspDidDynamicallyDefined = FALSE - DcmDspDataSize = 32 - DcmDspDataInfo.DcmDspDataFixedLength = TRUE - DcmDspDataType = UINT32 - DcmDspDataUsePort = USE_DATA_SENDER_RECEIVER - In the DcmDslProtocolRow used for test execution: - DcmDslProtocolEndiannessConvEnabled = TRUE - A sender/receiver interface “DataService_<Data>” for this DID has to be configured. A read wait point for the sender/receiver interface has to be configured for the SWC.	
Summary	Verify that the SUT correctly performs endianness conversion when writing and reading DID data: The LT writes a defined UINT32 value through UDS to the DID. The SWC receives this value and XORs it with the value 0x01020304. Then, LT reads the DID through UDS and the SWC returns the result of the XOR operation. The LT expects to receive “the original value XOR 0x04030201” because the SUT should have applied the endianness conversions for both the write and the read requests from the LT.	
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [N/A] Test Steps: [N/A] Feature under test is not available in R3.2.2.	
Pre-conditions	1. UDS connection between LT and SUT established 2. SWC and LT: Initialize “org” with a UINT32 value of which each byte is different to each other (e.g. 0x1234ABCD)	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC:** Wait at read wait point for incoming data	
Step 2	**LT:** Send UDS request “WriteDataByIdentifier” for DID=0xFE55 and with value “org” to SUT	
Step 3	**SWC:** Receive data at read wait point and store it in “rx” [SWS_Dcm_00638]	
Step 4	**LT:** Receive UDS response from SUT	Positive response code is received
Step 5	**SWC:** Calculate “tx = rx XOR 0x01020304”	
Step 6	**SWC:** Send “tx” to the sender/receiver interface of DID=0xFE55	

Step 7	**LT:** Send UDS request “ReadDataByIdentifier” for DID=0xFE55 to SUT	
Step 8	**LT:** Receive UDS response from SUT and store received value in “val”	Positive response code is received
Step 9	**LT:** Compare “val” with “org”	“val” must be equal to “org XOR 0x04030201”
Post-conditions	None	

5.2.6 [ATS_DIAG_00027] Retrieving of scaling information

Test Objective	Retrieving of scaling information		
ID	ATS_DIAG_00027	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00258 DiagnosticCommunicationManager: SWS_Dcm_00394		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	The data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE66 - DcmDspDidInfo.DcmDspDidDynamicallyDefined = FALSE - DcmDspDataSize = 8 - DcmDspDataInfo.DcmDspDataFixedLength = TRUE - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - DcmDspDataInfo.DcmDspDataScalingInfoSize = 9 - Callbacks for “GetScalingInformation” mapped to runnable of the SWC		
Summary	Verify that retrieving of scaling information of a DID works correctly: When the SUT receives the diagnostic request for retrieving the scaling information of a DID, verify that the SWC function configured for “GetScalingInformation” is called and that the SUT sends the correct data in the response.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] — Some parameter names for DID configuration are different in R3.2.2. Test Steps: [none]		
Pre-conditions	1. UDS connection between LT and SUT established		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	**LT:** Send UDS request “ReadScalingDataByIdentifier” for DID=0xFE66 to SUT	
Step 2	**SWC:** In runnable configured as “GetScalingInformation” callback, return the hex data “01 90 00 E0 4B 00 1E A0 30” (taken from example in ISO 14229) as ScalingInfo, 0 as ErrorCode and E_OK as return value	Runnable configured as “GetScalingInformation” callback is called by the RTE [SWS_Dcm_00394]
Step 3	**LT:** Receive UDS response from SUT and store the received DID scaling information in “info”	Positive response code received
Step 4	**LT:** Verify the length and content of “info”	“info” has the length “9” and the hex data content “01 90 00 E0 4B 00 1E A0 30”
Post-conditions	None	

5.2.7 [ATS_DIAG_00028] Periodical reading of data

Test Objective	Periodical reading of data		
ID	ATS_DIAG_00028	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00437 DiagnosticCommunicationManager: SWS_Dcm_00254 DiagnosticCommunicationManager: SWS_Dcm_00395		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	The transmission of periodic responses are configured to use the normal diagnostic responses: - DcmDslProtocolTransType = TYPE1 - An additional protocol (DcmDslProtocolRow) with DcmDslProtocolID = DCM_PERIODIC_TRANS_ON_IP | DCM_PERIODIC_ON_CAN | DCM_PERIODIC_ON_FLEXRAY - DcmDslPeriodicTransmissionConRef = <Reference to additional protocol above> A periodic data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xF277 (inside periodical DID range) - DcmDspDidInfo.DcmDspDidDynamicallyDefined = FALSE - DcmDspDataSize = 32 - DcmDspDataInfo.DcmDspDataFixedLength = TRUE		

	- DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - Callbacks for “ReadData” and “WriteData” mapped to runnables of the SWC
Summary	Verify that the periodical reading of data works correctly: First, the periodical DID is written to with a random value. Then, when the SUT receives the diagnosis request for periodical reading the data of a DID, verify that the SWC function configured for “ReadData” is called periodically and that the SUT sends the correct data (which is changing) in all periodic responses for a specified time. After sending the diagnosis request for stopping the periodical read, the SUT must stop sending the periodic responses.
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Test Steps: [low] Some parameter names for DID configuration are different in R3.2.2. In R3.2.2, the callout "ConditionCheckWrite" is available (but not in R4.x) and should therefore be checked additionally after LT has sent the "WriteDataByIdentifier" request.
Pre-conditions	1. UDS connection between LT and SUT established 2. LT: The variable “rnd” is initialized with a random value of type UINT32

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**LT:** Send UDS request “WriteDataByIdentifier” for DID=0xF277 and with value “rnd” to SUT	
Step 2	**SWC:** In runnable configured as “WriteData” callback, store the received DID value to “val” , return 0 as error code and E_OK as return value	Runnable configured as “WriteData” callback is called by the RTE [Dcm395]
Step 3	**LT:** Receive UDS response from SUT	Positive response code is received
Step 4	**LT:** Send UDS request “ReadDataByPeriodicIdentifier” for periodic DID=0x77 (i.e. 0xF277) with transmissionMode=0x02 (sendAtMediumRate) to SUT	
Step 5	The following loop is repeated and terminated by LT sending a UDS request with “transmissionMode=stopSending”	
Step 6	**LOOP: SWC:** In runnable configured as “ReadData” callback, return the value “val” as DID data and E_OK as return value	Runnable configured as “ReadData” callback is called by the RTE [Dcm437]
Step 7	**LOOP: SWC:** Increment “val” by 1 (with wrap-around)	
Step 8	**LOOP: LT:** Receive UDS response from SUT and check received DID value	Positive response code is received. Received DID value must be equal to “rnd”
Step 9	**LOOP: LT:** Increment “rnd” by 1 (with wrap-around)	
Step 10	**LOOP: LT:** If the loop has not already been iterated for 10 times, go to the beginning of the loop	

Step 11	**LT:** Send UDS request “ReadDataByPeriodicIdentifier” for periodic DID=0x77 (i.e. 0xF277) with transmissionMode=0x04 (stopSending) to SUT	
Step 12	**LT:** Receive UDS response from SUT	Positive response code is received
Step 13	**LT:** Check for further UDS responses for service “ReadDataByPeriodicIdentifier” from SUT	No UDS responses for service “ReadDataByPeriodicIdentifier” are received anymore
Post-conditions	None	

5.2.8 [ATS_DIAG_00029] Invocation of callbacks for “InputOutputControlByIdentifier”

Test Objective	Invocation of callbacks for “InputOutputControlByIdentifier”		
ID	ATS_DIAG_00029	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00256 DiagnosticCommunicationManager: SWS_Dcm_00579 DiagnosticCommunicationManager: SWS_Dcm_00396 DiagnosticCommunicationManager: SWS_Dcm_00397 DiagnosticCommunicationManager: SWS_Dcm_00398 DiagnosticCommunicationManager: SWS_Dcm_00399		
Requirements / Reference to Test Environment	SWC: A tester SWC that observes the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	The data record identifier (DID) with the following properties must be configured: - DcmDspDidIdentifier = 0xFE88 - Contains one signal - DcmDspDataSize = 32 - DcmDspDataUsePort = USE_DATA_SYNCH_CLIENT_SERVER - DcmDspDidFreezeCurrentState = TRUE - DcmDspDidResetToDefault = TRUE - DcmDspDidReturnControlToEcu = TRUE - DcmDspDidShortTermAdjustment = TRUE - Each callback for “ShortTermAdjustment”, “FreezeCurrentState”, “ReturnControlToECU” and “ResetToDefault” of UDS service “InputOutputControlByIdentifier” mapped to a runnable of the SWC - RAM of appropriate size assigned Furthermore, a runnable to serve the Xxx_ReadData() callout for DID=0xFE88 must be implemented and mapped. This runnable reads data from the RAM assigned to the DID and returns it.		

Summary	Verify that the IOControl requests are correctly passed to the SWC: When the SUT receives each of the IOControl requests “ShortTermAdjustment”, “FreezeCurrentState”, “ReturnControlToECU” and “ResetToDefault”, verify that the SWC runnable configured for the specific request is correctly called.	
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Test Steps: [none] Some parameter names for DID configuration are different in R3.2.2.	
Pre-conditions	1. UDS connection between LT and SUT established	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**SWC:** Write 0x1A1B1C1D into memory of DID=0xFE88	
Step 2	**LT:** Send UDS request “InputOutputControlByIdentifier” for DID=0xFE88 and with "ControlOptionRecord“ = 0x03AABBCCDD, i.e. first byte = "InputOutputControlParameter” = 0x03 (ShortTermAdjustment) to SUT	**SWC:** Runnable configured as “ShortTermAdustment” callback is called by the RTE [SWS_Dcm_00399]
Step 3	**SWC:** In runnable configured as “ShortTermAdjustment” callback, check input parameter "ControlOptionRecord", then return 0x00 as ErrorCode and E_OK as return value	"ControlOptionRecord" must contain 0x03AABBCCDD
Step 4	**LT:** Receive UDS response for “InputOutputControlByIdentifier” from SUT and check response code and "ControlStatusRecord" of the response	- Response code indicates no error - "ControlStatusRecord" contains 0x031A1B1C1D
Step 5	**SWC:** Write 0x2A2B2C2D into memory of DID=0xFE88	
Step 6	**LT:** Send UDS request “InputOutputControlByIdentifier” for DID=0xFE88 and with “InputOutputControlParameter” set to 0x02 (FreezeCurrentState) to SUT	**SWC:** Runnable configured as “FreezeCurrentState” callback is called by the RTE [SWS_Dcm_00398]
Step 7	**SWC:** In runnable configured as “FreezeCurrentState” callback, return 0x00 as ErrorCode and E_OK as return value	
Step 8	**LT:** Receive UDS response for “InputOutputControlByIdentifier” from SUT and check response code and "ControlStatusRecord" of the response	- Response code indicates no error - "ControlStatusRecord" contains 0x022A2B2C2D
Step 9	**SWC:** Write 0x3A3B3C3D into memory of DID=0xFE88	
Step 10	**LT:** Send UDS request “InputOutputControlByIdentifier” for DID=0xFE88 and with “InputOutputControlParameter” set to 0x00 (ReturnControlToECU) to SUT	**SWC:** Runnable configured as “ReturnControlToECU” callback is called by the RTE [SWS_Dcm_00396]

Step 11	**SWC:** In runnable configured as “ReturnControlToECU” callback, return 0x00 as ErrorCode and E_OK as return value	
Step 12	**LT:** Receive UDS response for “InputOutputControlByIdentifier” from SUT and check response code and "ControlStatusRecord" of the response	- Response code indicates no error - "ControlStatusRecord" contains 0x003A3B3C3D
Step 13	**SWC:** Write 0x4A4B4C4D into memory of DID=0xFE88	
Step 14	**LT:** Send UDS request “InputOutputControlByIdentifier” for DID=0xFE88 and with “InputOutputControlParameter” set to 0x01 (ResetToDefault) to SUT	**SWC:** Runnable configured as “ResetToDefault” callback is called by the RTE [SWS_Dcm_00397]
Step 15	**SWC:** In runnable configured as “ResetToDefault” callback, return 0x00 as ErrorCode and E_OK as return value.	
Step 16	**LT:** Receive UDS response for “InputOutputControlByIdentifier” from SUT and check response code and "ControlStatusRecord" of the response	- Response code indicates no error - "ControlStatusRecord" contains 0x014A4B4C4D
Post-conditions	None	

5.3 Test Cases on RoutineServices (Dcm)

5.3.1 [ATS_DIAG_00030] Handling of RoutineControl with fixed-length data

Test Objective	Handling of RoutineControl with fixed-length data		
ID	ATS_DIAG_00030	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00257 DiagnosticCommunicationManager: SWS_Dcm_00400 DiagnosticCommunicationManager: SWS_Dcm_00401 DiagnosticCommunicationManager: SWS_Dcm_00402 DiagnosticCommunicationManager: SWS_Dcm_00403 DiagnosticCommunicationManager: SWS_Dcm_00404 DiagnosticCommunicationManager: SWS_Dcm_00405		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	A routine identifier (RID) with the following properties(in DcmDspRoutine) must be configured: - DcmDspRoutineIdentifier = 0x0200 - DcmDspRoutineUsed = TRUE - DcmDspRoutineFixedLength = TRUE - DcmDspStopRoutineSupported = TRUE - DcmDspRequestResultsRoutineSupported = TRUE - DcmDspRoutineInfoRef->DcmDspStartRoutineIn configured as 1 UINT8 signal - DcmDspRoutineInfoRef->DcmDspStartRoutineOut configured as 2 UINT8 signals - DcmDspRoutineInfoRef->DcmDspRoutineStopIn configured as 3 UINT8 signals - DcmDspRoutineInfoRef->DcmDspRoutineStopOut configured as 4 UINT8 signals - DcmDspRoutineInfoRef->DcmDspRoutineRequestResOut configured as 5 UINT8 signals - DcmDspRoutineUsePort = TRUE - Callouts for "StartRoutine", "StopRoutine" and "RequestResultsRoutine" mapped to appropriate runnables of the SWC (referred to by RUN_StartRoutine, RUN_StopRoutine, RUN_RequestResults).		
Summary	Verify that starting, stopping and requesting results for a routine with fixed-length data works correctly: The test system starts and stops the configured routine and requests its result. During these operations the size and the content of the passed data is checked.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [low] Parameter names for configuration of routines (container DcmDspRoutine) have changed between R3.2 and R4.0. Test Steps: [none]		

Pre-conditions	1. UDS connection between LT and SUT established	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**LT:** Send UDS request “RoutineControl” with sub-function “StartRoutine”, RID=0x0200 and RoutineControlOptionRecord=”0xA0”	
Step 2	**SWC:** Await invocation of “RUN_StartRoutine” callback [SWS_Dcm_00400]	Callback “RUN_StartRoutine” is called by the RTE
Step 3	**SWC:** In “RUN_StartRoutine”, check the received argument “dataIn1” [SWS_Dcm_00400]	“dataIn1” is equal to 0xA0
Step 4	**SWC:** In “RUN_StartRoutine”, return “dataOut1”=0xA1, “dataOut2”=0xA2, 0 as error code and E_OK as return value	
Step 5	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00401]	Positive response code received. “RoutineStatusRecord” is of size 2 and has the values “0xA1 0xA2”
Step 6	**LT:** Send UDS request “RoutineControl” with sub-function “StopRoutine”, RID=0x0200 and RoutineControlOptionRecord=”0xB0 0xB1 0xB2”	
Step 7	**SWC:** Await invocation of “RUN_StopRoutine” callback [SWS_Dcm_00402]	Callback “RUN_StopRoutine” is called by the RTE
Step 8	**SWC:** In “RUN_StopRoutine”, check the received arguments “dataIn1”, “dataIn2” and “dataIn3” [SWS_Dcm_00402].	“dataIn1” is equal to 0xB0, “dataIn2” is equal to 0xB1, “dataIn3” is equal to 0xB2
Step 9	**SWC:** In “RUN_StopRoutine”, return “dataOut1”=0xB3, “dataOut2”=0xB4, “dataOut3”=0xB5, “dataOut4”=0xB6, 0 as error code and E_OK as return value	
Step 10	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00403]	Positive response code received. “RoutineStatusRecord” is of size 4 and has the values “0xB3 0xB4 0xB5 0xB6”
Step 11	**LT:** Send UDS request “RoutineControl” with sub-function “RequestRoutineResults” and RID=0x0200	
Step 12	**SWC:** Await invocation of “RUN_RequestResults” callback [SWS_Dcm_00404]	Callback “RUN_RequestResults” is called by the RTE
Step 13	**SWC:** In “RUN_RequestResults”, return “dataOut1”=0xC0, “dataOut2”=0xC1, “dataOut3”=0xC2, “dataOut4”=0xC3, “dataOut5”=0xC4, 0 as error code and E_OK as return value	
Step 14	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00405]	Positive response code received. “RoutineStatusRecord” is of size 5 and has the values “0xC0 0xC1 0xC2 0xC3 0xC4”

Post-conditions	None

5.3.2 [ATS_DIAG_00031] Handling of RoutineControl with dynamic-length data

Test Objective	Handling of RoutineControl with dynamic-length data		
ID	ATS_DIAG_00031	**AUTOSAR Releases**	4.0.3|4.1.1
Affected Modules	Dcm	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00019		
Trace to R4.1.1 Item	DiagnosticCommunicationManager: SWS_Dcm_00257 DiagnosticCommunicationManager: SWS_Dcm_00400 DiagnosticCommunicationManager: SWS_Dcm_00401 DiagnosticCommunicationManager: SWS_Dcm_00402 DiagnosticCommunicationManager: SWS_Dcm_00403 DiagnosticCommunicationManager: SWS_Dcm_00404 DiagnosticCommunicationManager: SWS_Dcm_00405		
Requirements / Reference to Test Environment	SWC: A tester SWC that interacts with the RTE interface of the SUT LT: Lower Tester that is capable of requesting UDS services on the SUT and receiving the UDS responses from the SUT		
Configuration Parameters	A routine identifier (RID) with the following properties(in DcmDspRoutine) must be configured: - DcmDspRoutineIdentifier = 0x0201 - DcmDspRoutineUsed = TRUE - DcmDspRoutineFixedLength = FALSE - DcmDspStopRoutineSupported = TRUE - DcmDspRequestResultsRoutineSupported = TRUE - DcmDspRoutineUsePort = TRUE - Callouts for "StartRoutine", "StopRoutine" and "RequestResultsRoutine" mapped to appropriate runnables of the SWC (referred to by RUN_StartRoutine, RUN_StopRoutine, RUN_RequestResults).		
Summary	Verify that starting, stopping and requesting results for a routine with dynamic-length data works correctly: The test system starts and stops the configured routine and requests its result. During these operations, the size and the content of the passed data is checked.		
Needed Adaptation to other Releases	**Needed Adaptation for Release 3.2.2** Configuration: [N/A] Test Steps: [N/A] — Feature under test is not available in R3.2.2		
Pre-conditions	1. The sizes N1, N2, …, N5 (each of value 1..65535 bits) for the dynamic lengths have been defined. 2. Arbitrary data contents for "DATA1 of length N1", "DATA2 of length N2" etc. (up		

	to "DATA5") have been defined. These data contents shall be byte-wise different from each other.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**LT:** Send UDS request “RoutineControl” with sub-function “StartRoutine”, RID=0x0201 and RoutineControlOptionRecord=”DATA1”	
Step 2	**SWC:** Await invocation of “RUN_StartRoutine” callback [SWS_Dcm_00400]	Callback “RUN_StartRoutine” is called by the RTE
Step 3	**SWC:** In “RUN_StartRoutine”, check the received arguments “currentDataLength” and “dataIn1”	“currentDataLength” is equal to N1, “dataIn1” is equal to “DATA1”
Step 4	**SWC:** In “RUN_StartRoutine”, return “dataOut1” = “DATA2”, "currentDataLength" = N2, 0 as error code and E_OK as return value	
Step 5	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00401]	Positive response code received. “RoutineStatusRecord” is of size N2 and has the content equal to “DATA2”
Step 6	**LT:** Send UDS request “RoutineControl” with sub-function “StopRoutine”, RID=0x0201 and RoutineControlOptionRecord=”DATA3”	
Step 7	**SWC:** Await invocation of “RUN_StopRoutine” callback [SWS_Dcm_00402]	Callback “RUN_StopRoutine” is called by the RTE
Step 8	**SWC:** In “RUN_StopRoutine”, check the received arguments “currentDataLength” and “dataIn1” [SWS_Dcm_00402]	“currentDataLength” is equal to N3, “dataIn1” is equal to “DATA3”
Step 9	**SWC:** In “RUN_StopRoutine”, return “dataOut1”=”DATA4”, "currentDataLength" = N4, 0 as error code and E_OK as return value	
Step 10	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00403]	Positive response code received. “RoutineStatusRecord” is of size N4 and has the content equal to “DATA4”
Step 11	**LT:** Send UDS request “RoutineControl” with sub-function “RequestRoutineResults” and RID=0x0201	
Step 12	**SWC:** Await invocation of “RUN_RequestResults” callback [SWS_Dcm_00404]	Callback “RUN_RequestResults” is called by the RTE
Step 13	**SWC:** In “RUN_RequestResults”, return “dataOut1=”DATA5”, "currentDataLength" = N5, 0 as error code and E_OK as return value	
Step 14	**LT:** Receive UDS response for service “RoutineControl” from SUT and check the received “RoutineStatusRecord” [SWS_Dcm_00405]	Positive response code received. “RoutineStatusRecord” is of size N5 and has the content equal to “DATA5”

Post-conditions	None