Document Title	Acceptance Test Specification of RTE
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	634
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Standard	Acceptance Tests for Classic Platform
Part of Standard Release	1.2.0

Document Change History

Date	Release	Changed by	Change Description
2016-12-15	1.2.0	AUTOSAR Release Management	• Checked and adapted to Classic Platform Release 4.2.2 • Added test suites for RS_BRF_01416 – RTE NvDataHandling • Added testcases for Rte Client Server Communication (ATS_RTE_00864 and ATS_RTE_00852) • Updated testcases for Sender-receiver communication (ATS_RTE_00654, ATS_RTE_00698 and ATS_RTE_00689)

2015-10-31	1.1.0	AUTOSAR Release Management	• Adaptations needed to test Classic Platform R3.2 are no more maintained. Please refer to Classic Platform Acceptance Tests R1.0.0.
			• Added test cases for
			  ○ Sender-receiver communication
			  ○ Scheduling (ATS_RTE_00694 & ATS_RTE_00707)
			  ○ Miscellaneous features (IRV, Enhanced modes, Ports, Pim, CData, Prm APIs)
			• Checked and adapted to Classic Platform Release 4.2.1
			• Formalization of the point of control and observation for actions and expected results
			• Formal changes

2014-07-30	1.0.0	AUTOSAR Release Management	• Initial release

Disclaimer

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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
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 all the RTE features.

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.

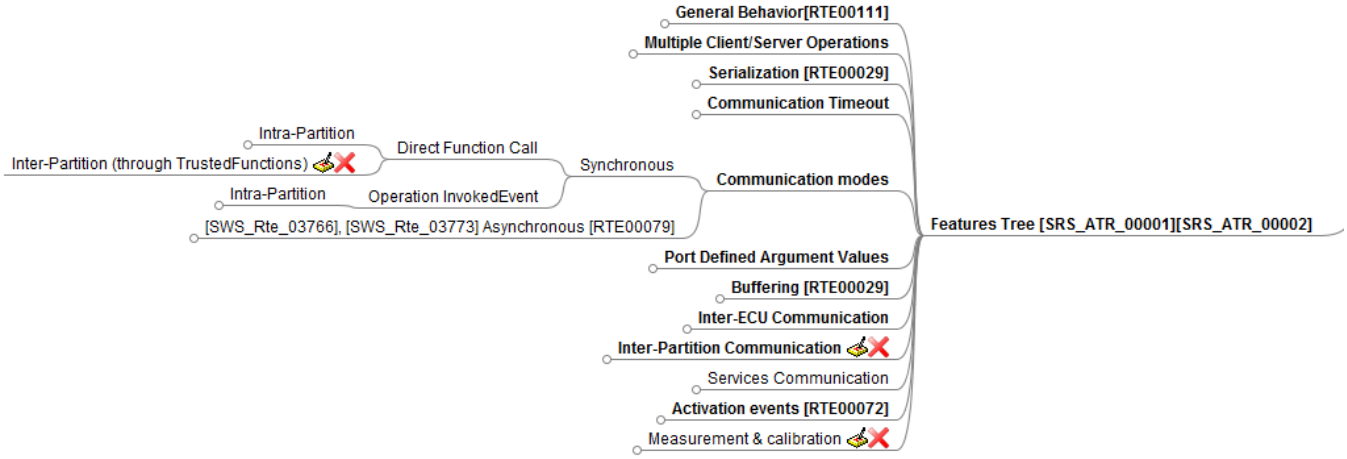
3 RS_BRF_01312 - Rte Client Server Communication

3.1 General Test Objective and Approach

This Test Specification covers the Client Server feature of the RTE as described in the AUTOSAR Feature [RS_BRF_01312].

The tests use a test bench environment and embedded Software Components that use the feature.

This test specification document has been established to cover the following features:

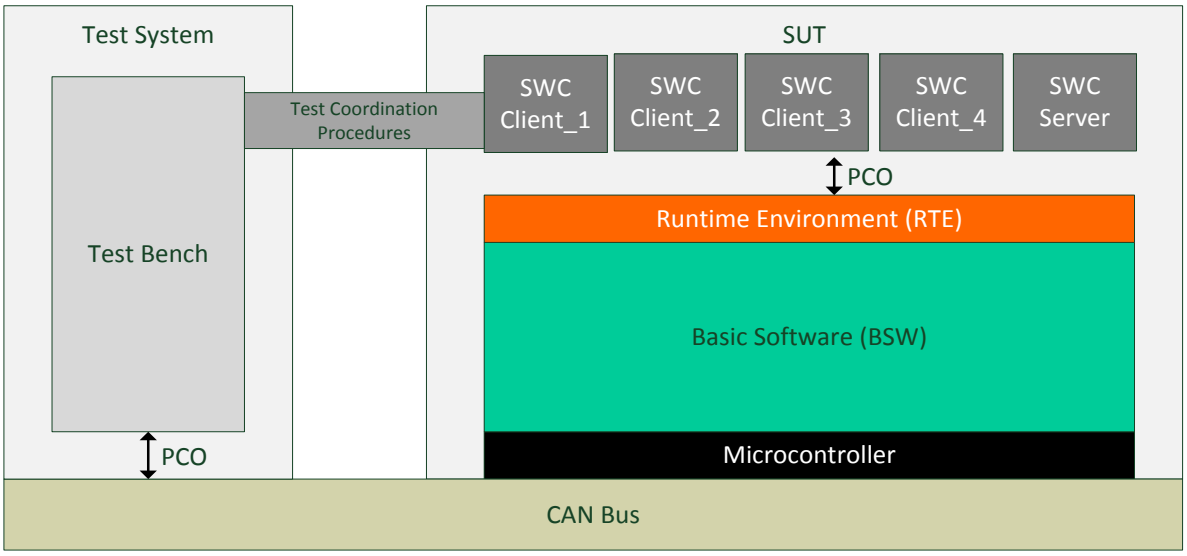


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

3.1.1 Test System

3.1.1.1 Overview on Architecture

In order to cover the required features / sub-features coverage, the environment has been separated in several use cases.

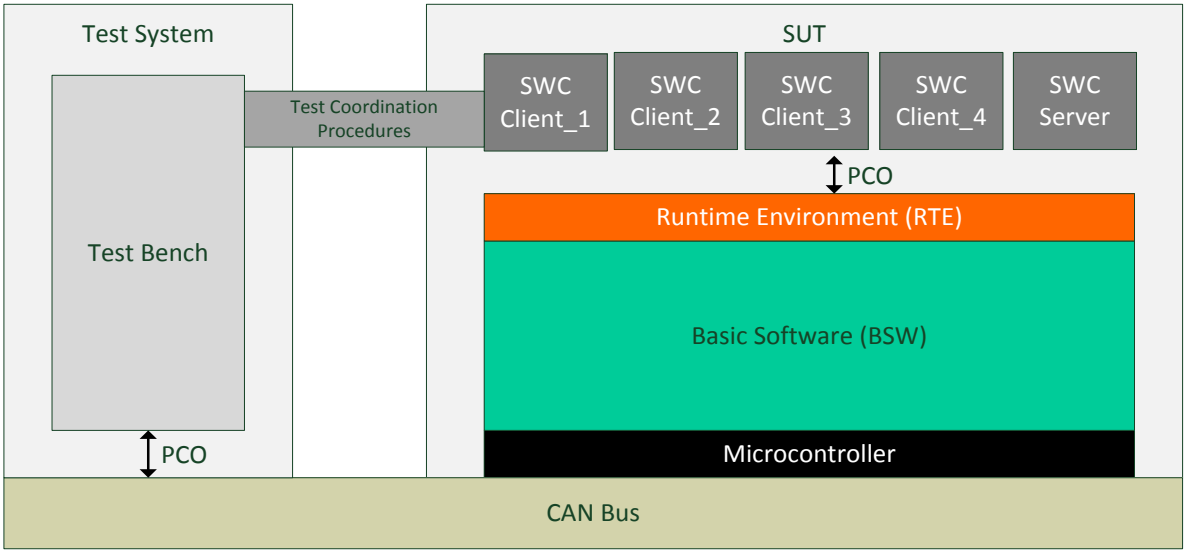


The test system architecture consists of Test Bench that executes only test sequencer and gives action requests through Test Coordination Procedures to embedded SWCs.

The location of SWC Clients and Server depends on the selected use case. The requirements for ECU Extract are described individually for each Use Case.

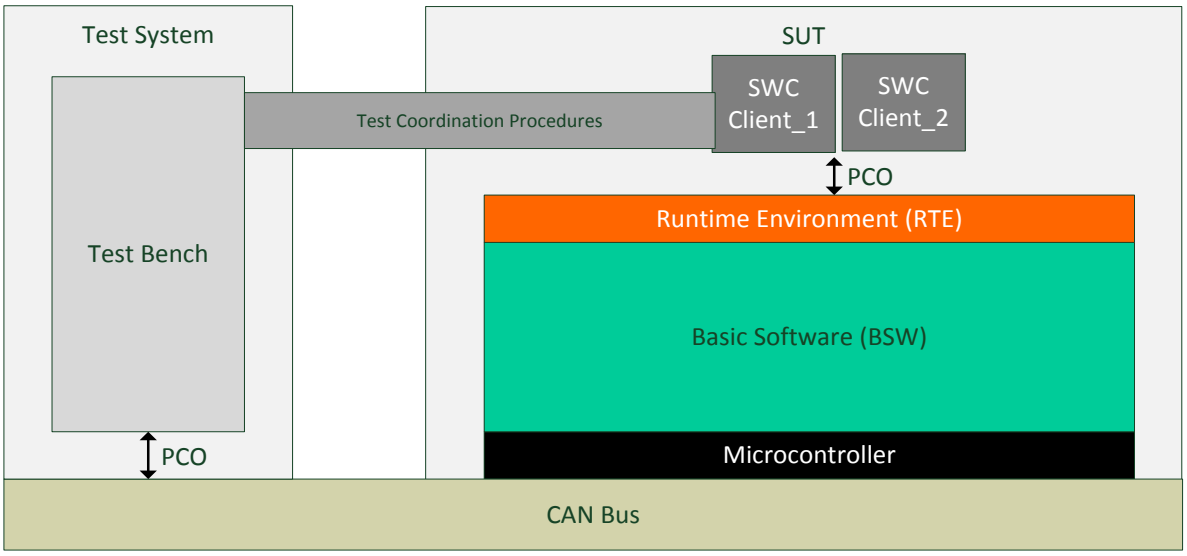
3.1.1.1.1 Use case UC01.01: Client-Server communication Intra-ECU

For this use case, the intention is to cover all intra-ECU communication (synchronous or asynchronous calls) inside an ECU.



3.1.1.1.2 Use case UC01.02.01: Client-Server communication Inter-ECU distant server

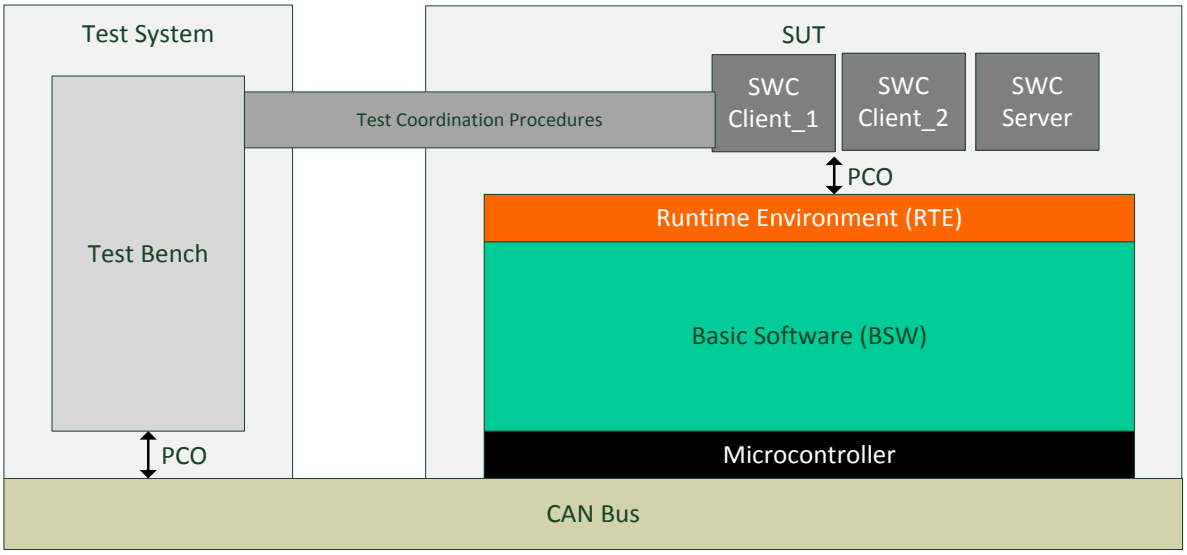
For this use case, the objective is to test the communication features of Clients to distant server as defined below:



In this use case, the test bench simulates the communication from clients to server.

3.1.1.1.3 Use case UC01.02.02: Client-Server communication Inter-ECU local server

For this use case, the objective is to test the communication features of Clients to local server and from distant clients to local server as defined below:



In this use case, the test bench simulates the communication from local server to distant clients.

3.1.1.2 Specific Requirements

Not Applicable

3.1.1.3 Test Coordination Requirements

Not Applicable

3.1.2 Test Configuration

This section describes sets of requirements on configuration.
These sets are later referenced by test cases.
No configuration files are provided. They need to be developed when the test suite is implemented.

3.1.2.1 Required ECU Extract of System Description Files

3.1.2.1.1 Generic SWC description

The SWC description has been established to cover all use cases. Only the mapping of com signals and ECU Extract is specific.

The SWC requirements are described below:

3.1.2.1.1.1 ClientServerInterfaces

Name	Operation	Arguments type
PrimitiveData_IF	Write	uint8 Data : IN
	Read	uint8 Data : OUT
	ReadWrite	uint8 Data : INOUT
Repeat_IF	Repeat	uint8 Data : IN
		uint8 Data : OUT
	RepeatArray	uint8[3] Data : IN
		uint8[3] Data : OUT
Reentrant_IF	Reentrant	uint8 Data : IN

3.1.2.1.1.2 SWC Tester_Client_1

SWC Name	Tester_Client_1		
PORTS	Name	Client1A	
	Type	RPortPrototype	
	Interface	PrimitiveData_IF	
	Requirements		
	Name	Reentrant1	
	Type	RPortPrototype	
	Interface	Reentrant_IF	
	Requirements		
RUNNABLE ENTITIES	Name	RUN_Client1	
	Requirements		
	ServerCallPoints	Name	sscp_Read1A
		Type	SynchronousServerCallPoint
		Access to	Client1A (Read operation)

		Requirements	
		Name	sscp_Write1A
		Type	SynchronousServerCallPoint
		Access to	Client1A (Write operation)
		Requirements	
		Name	sscp_ReadWrite1A
		Type	SynchronousServerCallPoint
		Access to	Client1A (Read and Write operation)
		Requirements	
		Name	sscp_Reentrant1
		Type	SynchronousServerCallPoint
		Access to	Reentrant1 (Reentrant operation)
		Requirements	

3.1.2.1.1.3 SWC Tester_Client_2

SWC Name	Tester_Client_2	
PORTS	**Name**	Client2A
	Type	RPortPrototype
	Interface	PrimitiveData_IF
	Requirements	
	Name	Client2B
	Type	RPortPrototype
	Interface	PrimitiveData_IF
	Requirements	
	Name	Client2C
	Type	RPortPrototype
	Interface	PrimitiveData_IF
	Requirements	
	Name	RUN_Client2
	Requirements	

RUNNABLE ENTITIES	**ServerCallPoints**	**Name**	ascp_Read2A
		Type	AsynchronousServerCallPoint
		Access to	Client2A (Read operation)
		Requirements	
		Name	ascp_Write2A
		Type	AsynchronousServerCallPoint
		Access to	Client2A (Write operation)
		Requirements	
		Name	ascp_ReadWrite2A
		Type	AsynchronousServerCallPoint
		Access to	Client2A (ReadWrite operation)
		Requirements	
		Name	ascp_Read2B
		Type	AsynchronousServerCallPoint
		Access to	Client2B (Read operation)
		Requirements	
		Name	ascp_Write2B
		Type	AsynchronousServerCallPoint
		Access to	Client2B (Write operation)
		Requirements	
		Name	ascp_Write2C
		Type	AsynchronousServerCallPoint
		Access to	Client2C (Write operation)
		Requirements	
	Name	RUN_Results2	
	Requirements	Shall be started by AsynchronousServerCallReturnsEvents (ASCRE_Read2A, ASCRE_Write2A, ASCRE_ReadWrite2A, ASCRE_Read2B, ASCRE_Write2B) which refer to ascrp_Read2A, ascrp_Write2A, ascrp_ReadWrite2A, ascrp_Read2B, ascrp_Write2B	
	AsynchronousServerCallResultPoint	**Name**	ascrp_Read2A
		Type	AsynchronousServerCallResultPoint

		Access to	ascp_Read2A
		Requirements	
		Name	ascrp_Write2A
		Type	AsynchronousServerCallResultPoint
		Access to	ascp_Write2A
		Requirements	
		Name	ascrp_ReadWrite2A
		Type	AsynchronousServerCallResultPoint
		Access to	ascp_ReadWrite2A
		Requirements	
		Name	ascrp_Read2B
		Type	AsynchronousServerCallResultPoint
		Access to	ascp_Read2B
		Requirements	
		Name	ascrp_Write2B
		Type	AsynchronousServerCallResultPoint
		Access to	ascp_Write2B
		Requirements	

3.1.2.1.1.4 SWC Tester_Client_3

SWC Name	Tester_Client_3			
PORTS	**Name**	Client3A		
	Type	RPortPrototype		
	Interface	PrimitiveData_IF		
	Requirements			
	Name	Client3B		
	Type	RPortPrototype		
	Interface	PrimitiveData_IF		
	Requirements			
	Name	Reentrant3		
	Type	RPortPrototype		
	Interface	Reentrant_IF		
	Requirements			
RUNNABLE ENTITIES	**Name**	RUN_Client3		
	Requirements			
	ServerCallPoint	**Name**	sscp_Read3A	
		Type	SynchronousServerCallPoint	
		Access to	Client3A (Read operation)	
		Requirements		
		Name	sscp_Write3A	
		Type	SynchronousServerCallPoint	
		Access to	Client3A (Write operation)	
		Requirements		
		Name	sscp_Read3B	
		Type	SynchronousServerCallPoint	
		Access to	Client3A (Read operation)	
		Requirements		
		Name	sscp_Write3B	
		Type	SynchronousServerCallPoint	
		Access to	Client3A (Write operation)	

		Requirements	
		Name	sscp_Reentrant3
		Type	SynchronousServerCallPoint
		Access to	Reentrant3 (Reentrant operation)
		Requirements	

3.1.2.1.1.5 SWC Tester_Client_4

SWC Name	Tester_Client_4		
	Name	Repeat4	
	Type	RPortPrototype	
	Interface	Repeat_IF	
	Requirements		
	Name	Client4B	
	Type	RPortPrototype	
	Interface	PrimitiveData_IF	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_Client4	
	Requirements		
	ServerCallAccesspoint	**Name**	sscp_Repeat4
		Type	SynchronousServerCallPoint
		Access to	Repeat4 (Repeat operation)
		Requirements	
		Name	sscp_RepeatArray4
		Type	SynchronousServerCallPoint
		Access to	Repeat4 (RepeatArray operation)
		Requirements	
		Name	sscp_Write4B
		Type	SynchronousServerCallPoint
		Access to	Client4B (Write operation)
		Requirements	

3.1.2.1.1.6 SWC Tester_Server

SWC Name	Tester_Server		
PORTS	**Name**	ServerA	
	Type	PPortPrototype	
	Interface	PrimitiveData_IF	
	Requirements		
	Name	ServerB	
	Type	PPortPrototype	
	Interface	PrimitiveData_IF	
	Requirements		
	Name	RepeatS	
	Type	PPortPrototype	
	Interface	Repeat_IF	
	Requirements		
	Name	ReentrantS	
	Type	PPortPrototype	
	Interface	Reentrant_IF	
	Requirements		
RUNNABLE ENTITIES	**Name**	serverRead	
	Requirements	canBeInvokedConcurrently=false send back data from global variable written by serverWrite	
	Started by Event	**Name**	OIE_ReadA
		Type	OperationInvokedEvent
			Port: ServerA Operation: Read
		Requirements	
		Name	OIE_ReadB
		Type	OperationInvokedEvent
			Port: ServerB Operation: Read
		Requirements	
	Name	serverWrite	
	Requirements	canBeInvokedConcurrently=false store received data in global variable (see serverRead)	

	Started by Event	**Name**	OIE_WriteA
		Type	OperationInvokedEvent
			Port: ServerA Operation: Write
		Requirements	
		Name	OIE_WriteB
		Type	OperationInvokedEvent
			Port: ServerB Operation: Write
		Requirements	
	Name	serverReentrant	
	Requirements	canBeInvokedConcurrently=true	
	Started by Event	**Name**	OIE_Reentrant
		Type	OperationInvokedEvent
			Port: ReentrantS Operation: Reentrant
		Requirements	
	Name	serverRepeat	
	Requirements	Copy IN parameter to the OUT parameter	
	Started by Event	**Name**	OIE_Repeat
		Type	OperationInvokedEvent
			Port: RepeatS Operation: Repeat
		Requirements	
	Name	serverRepeatArray	
	Requirements	Copy IN parameter to the OUT parameter	
	Event	**Name**	OIE_RepeatArray
		Type	OperationInvokedEvent
			Port: RepeatS Operation: RepeatArray
		Requirements	
	Name	serverReadWrite	
	Requirements	canBeInvokedConcurrently=false	

		XOR The received data (INOUT parameter) with 0xFF and copy the result into INOUT parameter	
	Event	**Name**	OIE_ReadWrite
		Type	OperationInvokedEvent
			Port: ServerA Operation: ReadWrite
		Requirements	

3.1.2.1.2 Use case UC01.01: intra-ECU Client-Server communication

The objective of the use case 1 is to test the Client-Server **intra-ECU** communication. This means that for this configuration, all SWC Clients and server are located in the SUT as described in the figure below:

3.1.2.1.2.1 Use case UC01.01: intra-ECU Client-Server communication

The objective of the use case 1 is to test the Client-Server **intra-ECU** communication. This means that for this configuration, all SWC Clients and server are located in the SUT as described in the figure below:

For this use case, all the SW-Cs are mapped to the SUT ECU.

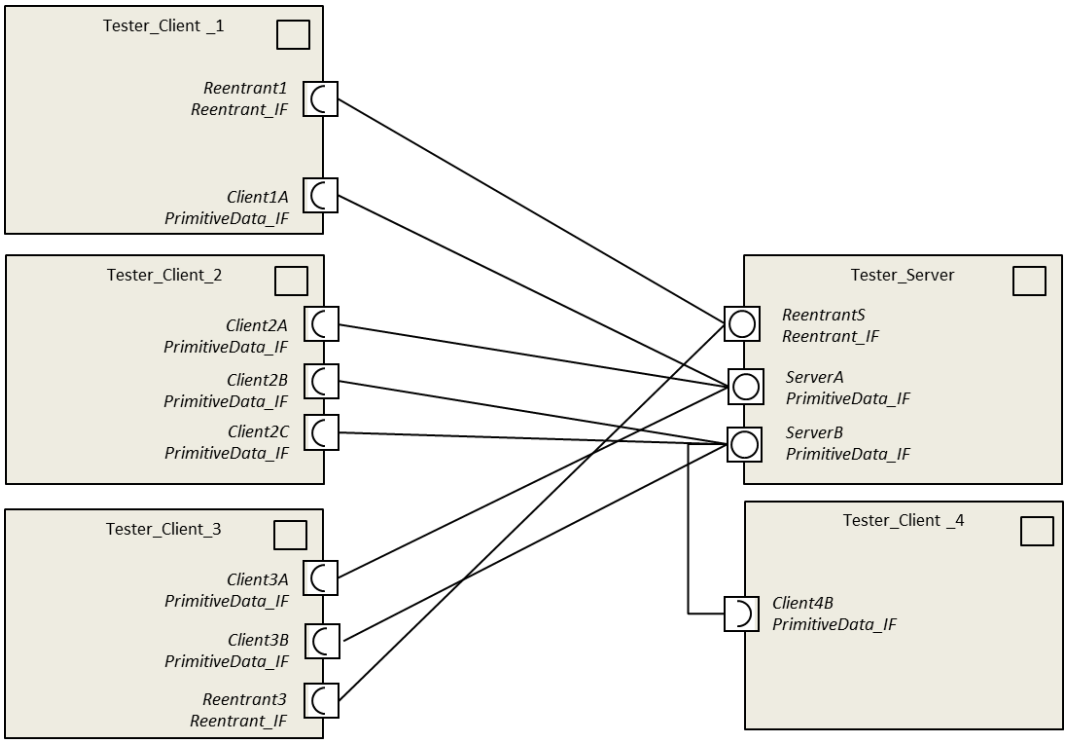


Figure 1: UC01.01: intra-ECU Client-Server communication

Some tests cases use Exclusive Area to not rely on OsTasks relative priorities.

The allocation of runnables to tasks is implementation specific.

3.1.2.1.3 Use case UC01.02.01: inter-ECU Client-Server communication – remote server

The objective of the use case 2.1 is to test the Client-Server **inter-ECU** communication **with distant server**. This means that for this configuration, only the SWC Tester_Client_1 and Tester_Client_2 are mapped on SUT.

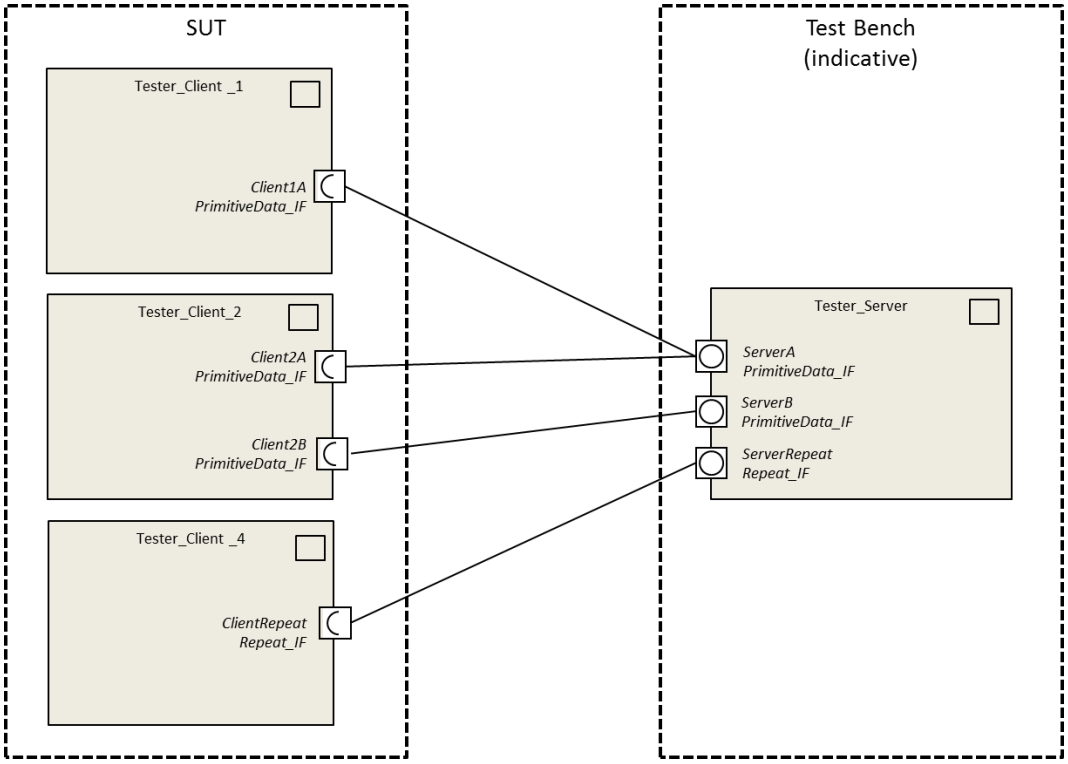


Figure 2: UC01.02.01: inter-ECU Client-Server communication – remote server

The following frames are used for the inter-ECU communication between clients and servers:

Frame	Description	Tx ECU	Rx ECU
WRITE_A_REQ	Contain the callSignal for the call of operation Write on ServerA. It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the IN parameter of the Write operation (uint8)	SUT	TestBench
WRITE_A_RES	Contain the returnSignal for the result of operation Write on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8)	TestBench	SUT

	• the return value (uint8)		
READ_A_REQ	Contain the callSignal for the call of operation Read on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8)	SUT	TestBench
READ_A_RES	Contain the returnSignal for the result of operation Read on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the read value (uint8) • the return value (uint8)	TestBench	SUT
REPEATARRAY_REQ	Contain the callSignal for the call of operation RepeatArray on ServerRepeat It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • 3 bytes for IN value (3 uint8 array elements)	TestBench	SUT
REPEATARRAY_RES	Contain the returnSignal for the result of operation RepeatArray on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • 3 bytes fort the OUT value (3 uint8 array elements) • the return value (uint8)	TestBench	SUT

3.1.2.1.4 Use case UC01.02.02: inter-ECU Client-Server communication – remote clients

The objective of the use case 2.2 is to test the Client-Server **inter-ECU** communication **with local server**. This means that for this configuration, only the SWC Tester_Client_1, Tester_Client_2 and Tester_Server are mapped on SUT.

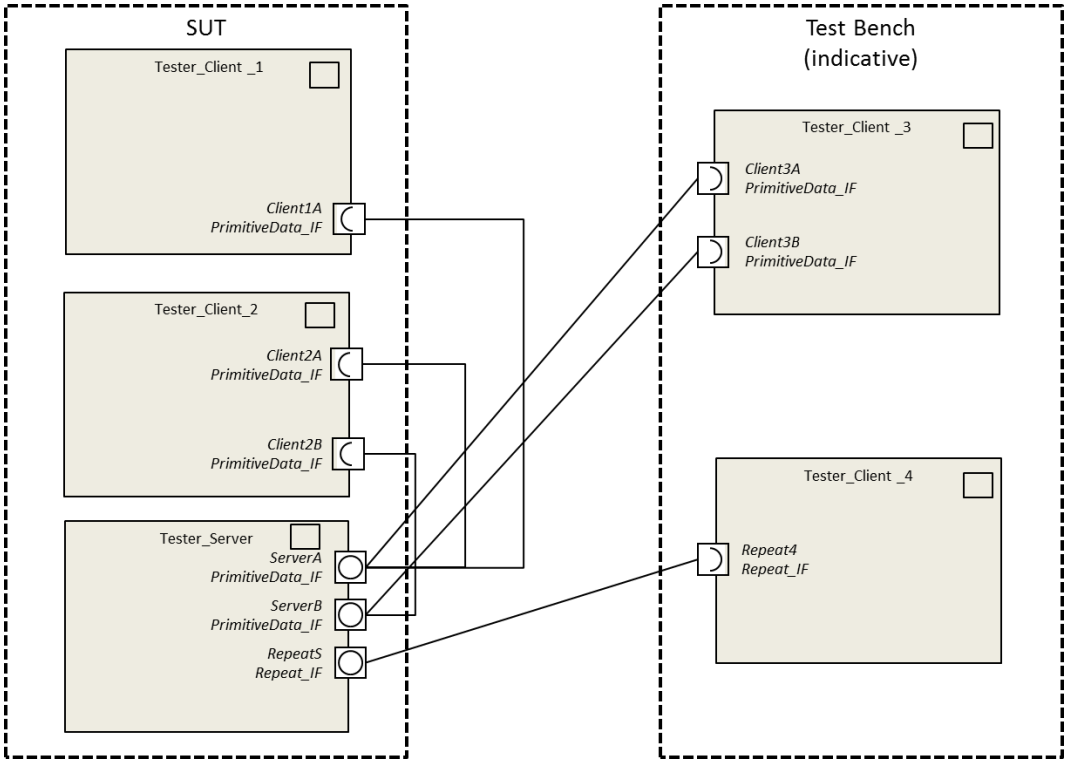


Figure 3: UC01.02.01: inter-ECU Client-Server communication – remote clients

The following frames are used for the inter-ECU communication between clients and servers:

Frame	Description	Tx ECU	Rx ECU
WRITE_A_REQ	Contain the callSignal for the call of operation Write on ServerA. It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the IN parameter of the Write operation (uint8)	TestBench	SUT
WRITE_A_RES	Contain the returnSignal for the result of operation Write on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the return value (uint8)	SUT	TestBench

READ_A_REQ	Contain the callSignal for the call of operation Read on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8)	TestBench	SUT
READ_A_RES	Contain the returnSignal for the result of operation Read on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the read value (uint8) • the return value (uint8)	SUT	TestBench
REPEATARRAY_REQ	Contain the callSignal for the call of operation RepeatArray on ServerRepeat It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • 3 bytes for IN value (3 uint8 array elements)	TestBench	SUT
REPEATARRAY_RES	Contain the returnSignal for the result of operation RepeatArray on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • 3 bytes fort the OUT value (3 uint8 array elements) • the return value (uint8)	SUT	TestBench
REPEATARRAY_REQ	Contain the callSignal for the call of operation Repeat on ServerRepeat It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the IN value (uint8)	TestBench	SUT
REPEAT_RES	Contain the returnSignal for the result of operation Repeat on ServerA It includes the following signals: • clientId (uint8) • sequenceCounter (uint8) • the OUT value (uint8) • the return value (uint8)	SUT	TestBench

3.1.2.2 Required ECU Configuration Description Files

There are no generic requirements on ECU Configuration files. Individual test cases may have specific requirements on the RTE configuration.

3.1.2.3 Required Software Component Description Files

Requirements on Software Component Description files are provided in the section 3.1.2.1 Required ECU Extract of System Description Files, together with the connections of the different components.

3.1.2.4 Mandatory vs. Customizable Parts

Not Applicable.

3.1.3 Test Case Design

Not Applicable.

3.2 Re-usable Test Steps

Not Applicable

3.3 Test Cases

3.3.1 [ATS_RTE_00052] Test synchronous server call for n:1 intra-ECU Client-Server communication

Test Objective	Test synchronous server call for n:1 intra-ECU Client-Server communication		
ID	ATS_RTE_00052	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_02527 RTE: SWS_Rte_04515 RTE: SWS_Rte_04516 RTE: SWS_Rte_04519		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Read1A, sscp_Write1A) Tester_Client_3 * port Client3A * runnable RUN_Client3 (sscp_Write3A) Tester_Server * port ServerA * runnable serverRead * runnable serverWrite Client1A and Client3A connected to ServerA. ServerA triggers serverRead (resp. serverWrite) for the Read (resp. Write) operation.		
Summary	The goal of this test is to check the behavior of synchronous server calls in case of n:1 Intra-ECU Client-Server communication. 2 clients connected to the same server are invoking (synchronous server call) successively the same operation of the server. The Test Manager checks that the operations are handled correctly.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start RUN_Client1, RUN_Client3		

Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Write(DataValue1)	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client3>]** execute Rte_Call_Client3A_Write(DataValue2)	**[RUN<RUN_Client3>]** Rte_Call returns RTE_E_OK
Step 4	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK, data returned is DataValue2
Step 5	**[CP]** terminate RUN_Client1, RUN_Client3	
Post-conditions	None	

3.3.2 [ATS_RTE_00054] Test intra-ECU C-S with operations of 2 ports mapped on one runnable (synchronous server call)

Test Objective	Test intra-ECU C-S with operations of 2 ports mapped on one runnable (synchronous server call)		
ID	ATS_RTE_00054	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_04520 RTE: SWS_Rte_04522 RTE: SWS_Rte_08001 RTE: SWS_Rte_08002		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Write1A) Tester_Client_3 * port Client3B * runnable RUN_Client3 (sscp_Write3B) Tester_Client_4 * port Client4B * runnable RUN_Client4 (sscp_Write4B) Tester_Server * port ServerA * port ServerB		

	* runnable serverWrite Client1A connected to ServerA. Client3B and Client4B connected to ServerB. Both ServerA and ServerB triggers serverWrite when the Write operation is invoked. ServerComSpec.queueLength set to 2 on both ServerA and ServerB for operation Write. serverWrite has its canBeInvokedConcurrently attribute set to false
Summary	The goal of this test is to check the behavior of synchronous server calls when 2 operations are mapped on the same runnable entity (on server side). 3 clients are calling the same operation, the clients are connected to the server on 2 different ports. The operations on those 2 ports (server side) start the same server runnable. The server has a queue length (ServerComSpec.queueLength) set to 2 on both ports. It needs to be ensured that the first requests are not processed before the third one is issued (see HINT below). The Test Manager checks that the first 2 requests are handled correctly by the server in the requested sequence, and that the third request results in a RTE_E_LIMIT error. HINT: For example, the clients runnables can be mapped to tasks with the following relative priorities: Prio RUN_Client4<Prio RUN_Client3<Prio RUN_Client1 The Task containing the Test Manager should have a priority higher than Clients The Task containing the Server should have a lower priority than Clients
Needed Adaptation to other Releases	
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client1, RUN_Client3, RUN_Client4	
Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Write	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client3>]** execute Rte_Call_Client3B_Write	**[RUN<RUN_Client3>]** Rte_Call returns RTE_E_OK
Step 4	**[RUN<RUN_Client4>]** execute Rte_Call_Client4B_Write	**[RUN<RUN_Client4>]** Rte_Call returns RTE_E_LIMIT
Step 5	**[CP]**	

	terminate RUN_Client1, RUN_Client3, RUN_Client4	
Post-conditions	None	

3.3.3 [ATS_RTE_00055] Test asynchronous server call for 1:1 intra-ECU Client-Server communication

Test Objective	Test asynchronous server call for 1:1 intra-ECU Client-Server communication		
ID	ATS_RTE_00055	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_03771		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Read2A, ascp_Write2A) * runnable RUN_Results2 (ascrp_Read2A, ASCRE_Read2A, ascrp_Write2A, ASCRE_Write2A) Tester_Server * port ServerA * runnable serverRead * runnable serverWrite Client2A connected to ServerA.		
Summary	The goal of this test is to check the behavior of asynchronous server calls in case of 1:1 intra-ECU Client-Server communication. This test involves communication between Tester_Client_2 and Tester_Server. The Tester_Client_2 SW-C calls the local server for the Read and Write operations. The Test Manager checks that asynchronous server calls for the Read and Write operations succeed (data is written and read correctly) and that an AsynchronousServerCallReturnsEvent is triggered to start a runnable of the client.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 4	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK, data returned is DataValueInit
Step 5	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write(DataValue1)	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 6	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by an AsynchronousServerCallReturnsEvent
Step 7	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK
Step 8	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 9	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by an AsynchronousServerCallReturnsEvent
Step 10	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK, data returned is DataValue1
Post-conditions	None	

3.3.4 [ATS_RTE_00056] Test asynchronous server call for n:1 intra-ECU Client-Server communication

Test Objective	Test asynchronous server call for n:1 intra-ECU Client-Server communication		
ID	ATS_RTE_00056	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement	ATR: ATR_ATR_00001		

on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_02527 RTE: SWS_Rte_02528 RTE: SWS_Rte_03773 RTE: SWS_Rte_04520
Requirements / Reference to Test Environment	UC01.01
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * port Client2C * runnable RUN_Client2 (ascp_Read2A, ascp_Write2A, ascp_Write2C) * runnable RUN_Results2 (ascrp_Read2A, ASCRE_Read2A, ascrp_Write2A, ASCRE_Write2A, ascrp_Write2C, ASCRE_Write2C) Tester_Server * port ServerA * runnable serverRead * runnable serverWrite Client2A and Client2C connected to ServerA. In addition, the system is configured so that the first Write request is not processed before the second one is issued. For example: * serverRead, serverWrite are mapped to tasks * serverRead and serverWrite are configured to run inside the same exclusive area EA1 * RUN_Client2 can enter this exclusive area EA1 (i.e. the RTE is configured to use the same OS object to implement the exclusive area of the server and client SW-C) * ServerA has a ServerComSpec for operation Write with queueLength 1
Summary	The goal of this test is to check the behavior of asynchronous server calls in case of n:1 intra ECU Client-Server communication. This test involves communication between Tester_Client_2 and Tester_Server. The Tester_Client_2 SWC calls the local server for Write operations on 2 different ports, both connected to the same server port. The Server has a queueLength = 1. The system is configured in such way that the first request is not processed before the second one is issued. The 2 requests as thus considered 'simultaneous'. The Test Manager shall check that the first asynchronous server call for the first operation succeeds (data is written and read correctly) and that an AsynchronousServerCallReturnsEvent is triggered to start a runnable of the client. The Test Manager shall check that the second (simultaneous) asynchronous server call for the Write operation (on the other port) results in a RTE_E_LIMIT error.

	HINT: To allow this behavior, the 2 simultaneous Client call to server operation should be encapsulated in an ExclusiveArea EA1 that is also used by the server. In this case, the server will not execute before call of the 2 successive requests.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Enter to enter EA1, execute Rte_Call_Client2A_Write	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 4	**[RUN<RUN_Client2>]** execute Rte_Call_Client2C_Write execute Rte_Exit to exit EA1	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 5	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK
Step 6	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK Read data is equal to the data written in step 2
Step 7	**[RUN<RUN_Results2>]** execute Rte_Result_Client2C_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_LIMIT
Step 8	**[CP]** terminate RUN_Client2, RUN_Results2	
Post-conditions	None	

3.3.5 [ATS_RTE_00057] Test intra-ECU C-S with operations of 2 ports mapped on one runnable (asynchronous server call)

Test Objective	Test intra-ECU C-S with operations of 2 ports mapped on one runnable (asynchronous server call)		
ID	ATS_RTE_00057	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001	
Trace to SWS Item	RTE: SWS_Rte_04520 RTE: SWS_Rte_04522 RTE: SWS_Rte_08001 RTE: SWS_Rte_08002	
Requirements / Reference to Test Environment	UC01.01	
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Read2A, ascp_Read2B) * runnable RUN_Results2 (ascrp_Read2A, ASCRE_Read2A, ascrp_Read2B, ASCRE_Read2B) Tester_Server * port ServerA * port ServerB * runnable serverRead Client2A connected to ServerA. Client2B connected to ServerB. Both ServerA and ServerB triggers serverRead when the Read operation is invoked. ServerComSpec.queueLength set to 1 on both ServerA and ServerB for operation Read. serverRead has its canBeInvokedConcurrently attribute set to false The server/system has to be designed/configured so that the first request is not processed before the second request is issued.	
Summary	The goal of this test is to check the behavior of asynchronous server calls when 2 operations are mapped on the same runnable entity (on server side). In this case, the server shall implement a single queue for the 2 operations (ServerComSpecs.queueLength = 1). The test manager has to ensure that the successive calls of the Read operation on 2 ports are issued before the first one is processed. The test manager checks that the first request is handled correctly and the second request results in a RTE_E_LIMIT error.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	

Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client2>]** execute Rte_Call_Client2B_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 4	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 5	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK
Step 6	**[RUN<RUN_Results2>]** execute Rte_Result_Client2B_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_LIMIT
Post-conditions	None	

3.3.6 [ATS_RTE_00058] Test intra-ECU Client-Server communication with timeout (asynchronous server call)

Test Objective	Test intra-ECU Client-Server communication with timeout (asynchronous server call)		
ID	ATS_RTE_00058	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_03764 RTE: SWS_Rte_03765 RTE: SWS_Rte_03766 RTE: SWS_Rte_03770 RTE: SWS_Rte_03771		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Write2A) * runnable RUN_Results2 (ascrp_Write2A, ASCRE_Write2A) Tester_Server * port ServerA * runnable serverWrite		

	Client2A connected to ServerA. In addition * Tester_Client_2 has a TriggerPort, accessible from RUN_Client2 * One Runnable RUN_BlockingProcess is started by a ExternalTriggeredOccurredEvent from TriggerPort Execution time of RUN_BlockingProcess higher than 10ms * ascp_Write2A configured with timeout = 10ms, and a WaitPoint for RUN_Results2 references the AsynchronousServerCallReturnsEvent for ascrp_Write2A	
Summary	The goal of this test case is to check the RTE timeout monitoring in case of an asynchronous server call for an intra-ECU Client-Server communication. This test uses Tester_Client_2, which calls a server of Tester_Server. This test is developed in a way that a timeout occurs. The test manager checks that a RTE_E_TIMEOUT error is provided to the client. HINT: In order to check the timeout monitoring of the RTE, this test case uses (in addition to the client and server) one external runnable that is triggered by Tester_Client_2 (ExternalTrigger). This external runnable has a higher priority than the server runnable, the client is mapped to a task with a higher priority than the external runnable. The execution time of the external runnable should be at least equal to the configured timeout.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write execute Rte_Trigger to trigger execution of RUN_BlockingProcess	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 4	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_TIMEOUT
Post-conditions	None	

3.3.7 [ATS_RTE_00059] Test asynchronous server call for 1:1 inter-ECU Client-Server communication

Test Objective	Test asynchronous server call for 1:1 inter-ECU Client-Server communication		
ID	ATS_RTE_00059	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item			
Requirements / Reference to Test Environment	UC01.02.01		
Configuration Parameters	See UC01.02.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Read2A, ascp_Write2A) * runnable RUN_Results2 (ascrp_Read2A, ASCRE_Read2A) Write request for Client2A transmitted on frame WRITE_A_REQ (includes IN parameter, clientId, sequenceCounter) Write result for Client2A transmitted on frame WRITE_A_RES (includes return value, clientId, sequenceCounter) Read request for Client2A transmitted on frame READ_A_REQ (includes IN parameter, clientId, sequenceCounter) Read result for Client2A transmitted on frame READ_A_RES (includes return value, OUT parameter, clientId, sequenceCounter)		
Summary	The goal of this test is to check the behavior of asynchronous server calls in case of 1:1 inter-ECU Client-Server communication. This test involves communication between Tester_Client_2 and a remote server (on test bench) Tester_Client_2 calls asynchronously the Write (IN parameter) operation of the remote server. The test manager checks that the request (including the IN parameter) is correctly transmitted on a bus. Then, Tester_Client_2 calls asynchronously the Read operation of the remote server. The test manager checks that • the Read request is correctly transmitted on a bus • the results of the Read operation, transmitted on a bus, is correctly provided to Tester_Client_2 (Rte_Result OUT parameter)		

Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write(DataValue1)	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** WAIT 100ms	**[LT]** frame WRITE_A_REQ with DataValue1 has been transmitted by SUT
Step 4	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 5	**[CP]** WAIT 100ms	**[LT]** frame READ_A_REQ has been transmitted by SUT
Step 6	**[LT]** send frame READ_A_RES with DataValue1 as OUT parameter, RTE_E_OK as return value, and both clientId/sequenceCounter as received in step 5	
Step 7	**[CP]** WAIT 100ms	
Step 8	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK Returned data value is equal to DataValue1
Post-conditions	None	

3.3.8 [ATS_RTE_00060] Test asynchronous server call for n:1 inter-ECU Client-Server communication

Test Objective	Test asynchronous server call for n:1 inter-ECU Client-Server communication		
ID	ATS_RTE_00060	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed

Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001
Trace to SWS Item	RTE: SWS_Rte_02649 RTE: SWS_Rte_02651
Requirements / Reference to Test Environment	UC01.02.01
Configuration Parameters	See UC01.02.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * port Client2B * runnable RUN_Client2 (ascp_Write2A, ascp_Write2B) * runnable RUN_Results2 (ascrp_Write2A, ASCRE_Write2A, ascrp_Write2A, ASCRE_Write2A) The Write requests on the 2 ports are mapped to the frame WRITE_A_REQ (for example both Client2A and Client2B are connected to serverA), with different client identifiers. The results of the Write requests on the 2 ports are mapped to the frame WRITE_A_RES.
Summary	The goal of this test is to check the behavior of asynchronous server calls in case of n:1 inter-ECU Client-Server communication. This test involves 2 clients (Tester_Client_2 with ports Client2A and Client2B), which are using asynchronous calls to a remote server on the test bench (for example Tester_Server). The 2 clients perform simultaneous requests to the server. Frames are sent from the test bench to provide a successful answer to the first request, and an RTE_E_LIMIT error to the second one. The test manager checks that the requests are correctly transmitted on a bus and that the results are correctly provided to the callers.
Needed Adaptation to other Releases	
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3		**[LT]** frame WRITE_A_REQ is transmitted on <Bus>

Step 4	**[RUN<RUN_Client2>]** execute Rte_Call_Client2B_Write	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 5		**[LT]** frame WRITE_A_REQ is transmitted on <Bus> (clientId differs from step 3)
Step 6	**[LT]** send frame WRITE_A_RES with the clientId and sequenceCounter received in step 3 and return value RTE_E_OK	
Step 7	**[LT]** send frame WRITE_A_RES with the clientId and sequenceCounter received in step 5 and return value RTE_E_LIMIT	
Step 8	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 9	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK
Step 10	**[RUN<RUN_Results2>]** execute Rte_Result_Client2B_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_LIMIT
Post-conditions	None	

3.3.9 [ATS_RTE_00061] Test inter-ECU C-S with operations of 2 ports mapped on one runnable (remote clients)

Test Objective	Test inter-ECU C-S with operations of 2 ports mapped on one runnable (remote clients)		
ID	ATS_RTE_00061	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_03769		
Requirements / Reference to Test Environment	UC01.02.02		
Configuration Parameters	See UC01.02.02 in chapter 3.1.2 Test Configuration. This test case uses:		

	Tester_Server * port ServerA * port ServerB * runnable serverWrite (invoked by the Write operation on both ports) frames WRITE_A_REQ, WRITE_A_RES, WRITE_B_REQ, WRITE_B_RES The server/system shall be implemented/configured in such way that the first request is not fully processed before the second request is issued.	
Summary	The goal of this test is to check the behavior of inter-ECU Client-Server communication when 2 operations are mapped on the same runnable entity (on server side). 2 server ports are mapped to the same server runnable, and are connected to remote clients. The servers are configured with a ServerComSpec.queueLength = 1. The Test Manager sends requests on the bus, and checks, on the bus, that the first request is handled correctly and the second request results in a RTE_E_LIMIT error.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Send frame WRITE_A_REQ with value DataValue1, a clientId and sequenceCounter	
Step 2	**[CP]** Send frame WRITE_B_REQ with value DataValue2, a clientId (different from the one of step 1), and a sequenceCounter	
Step 3	**[CP]** WAIT 100ms	**[RUN<serverWrite>]** serverWrite has been started once by an OperationInvokedEvent and received the value DataValue1
Step 4		**[LT]** WRITE_A_RES has been sent by SUT with return value RTE_E_OK and clientId/sequenceCounter as in step 1 WRITE_B_RES has been sent by SUT with return value RTE_E_LIMIT and clientId/sequenceCounter as in step 2
Post-conditions	None	

3.3.10 [ATS_RTE_00062] Test n:1 inter-ECU Client-Server communication with queue overflow

Test Objective	Test n:1 inter-ECU Client-Server communication with queue overflow		
ID	ATS_RTE_00062	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_04520 RTE: SWS_Rte_08002		
Requirements / Reference to Test Environment	UC01.02.02		
Configuration Parameters	See UC01.02.02 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Read1A) Tester_Server * port ServerA * runnable serverRead frames READ_A_REQ, READ_A_RES In addition: For ServerA: ServerComSpec.queueLength = 2 Client1A connected to ServerA. The server/system is implemented/configured so that the first (local) request is not fully processed before the 2 other requests are issued.		
Summary	The goal of this test is to check the behavior of n:1 inter-ECU, including the queue overflow. In this test case, multiple clients invoke the same operation of a server: a local client (synchronous server call) 2 remote clients (frames sent by the test manager, for example by Tester_Client_3 and Tester_Client_4 running on another AUTOSAR ECU) The server is configured with ServerComSpec.queueLength = 2 The test manager checks that the first remote request is processed successfully and the second request receives an RTE_E_LIMIT error.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]**		

	start RUN_Client1	
Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read (blocking call)	
Step 3	**[LT]** send frame READ_A_REQ with first clientId	
Step 4	**[LT]** send frame READ_A_REQ with second clientId	
Step 5	**[CP]** WAIT 100ms	**[RUN<RUN_Client1>]** Rte_Call has finished its execution, has returned RTE_E_OK and returned the value provided by the server
Step 6		**[LT]** 2 READ_A_RES frames have been transmitted: • one with the clienId/sequenceCounter of step 3, a return value set to RTE_E_OK and the value provided by the server • the other one with the clientId/sequenceCounter of step 4, and a return value set to RTE_E_LIMIT
Post-conditions	None	

3.3.11 [ATS_RTE_00063] Test inter-ECU Client-Server communication with timeout (synchronous server call)

Test Objective	Test inter-ECU Client-Server communication with timeout (synchronous server call)		
ID	ATS_RTE_00063	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_03763		
Requirements / Reference	UC01.02.01		

to Test Environment		
Configuration Parameters	See UC01.02.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Read1A) frame READ_A_REQ In addition: A timeout is configured on the sscp_Read1A: 10ms	
Summary	The goal of this test case is to check the behavior of inter-ECU Client-Server communication when a timeout occurs and the client uses a synchronous server call. For this test case, the server is on the test bench, the client (Tester_Client_1) performs a synchronous call to the server. The test bench does not provide a response within 20ms. The test manager checks that the client receives an RTE_E_TIMEOUT error. The client is able to perform another request to the server afterwards.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client1	
Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read	**[LT]** frame READ_A_REQ is transmitted on <Bus>
Step 3	**[CP]** WAIT 20ms (test bench remains silent on <Bus> during that time)	
Step 4	**[LT]** send frame READ_A_RES with the same clientId and sequenceCounter as in step 2 with value DataValue1	
Step 5		**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_TIMEOUT
Step 6	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read	**[LT]** frame READ_A_REQ is transmitted on <Bus> (same clientId, but different sequenceCounter than in step 2)

Step 7	**[LT]** send frame READ_A_RES with the same clientId and sequenceCounter as in step 6 with value DataValue2	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK with value DataValue2
Post-conditions	None	

3.3.12 [ATS_RTE_00064] Test inter-ECU Client-Server communication with timeout (asynchronous server call)

Test Objective	Test inter-ECU Client-Server communication with timeout (asynchronous server call)		
ID	ATS_RTE_00064	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_03763 RTE: SWS_Rte_03765 RTE: SWS_Rte_03772 RTE: SWS_Rte_03773		
Requirements / Reference to Test Environment	UC01.02.01		
Configuration Parameters	See UC01.02.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Read2A) * runnable RUN_Results2 (ascrp_Read2A, ASCRE_Read2A) frame READ_A_REQ In addition: A timeout is configured on ascp_Read2A: 10ms		
Summary	The goal of this test case is to check the behavior of inter-ECU Client-Server communication when a timeout occurs and the client uses an asynchronous server call. For this test case, the server is on the test bench, the client (Tester_Client_2) performs an asynchronous call to the server. The test bench does not provide a response within 20ms. The test manager checks that the client receives an RTE_E_TIMEOUT error. The client is able to perform another request to the server afterwards		

Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[LT]** frame READ_A_REQ transmitted on <Bus>
Step 3	**[CP]** WAIT 20ms (test bench remains silent during that time)	**[RUN<RUN_Results2>]** runnable has been started by an AsynchronousServerReturnsEvent
Step 4	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_TIMEOUT
Step 5	**[LT]** send frame READ_A_RES with the same clientId and sequenceCounter as in step 2 with value DataValue1	
Step 6	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Read	**[LT]** frame READ_A_REQ is transmitted on <Bus> (same clientId, but different sequenceCounter than in step 2)
Step 7	**[LT]** send frame READ_A_RES with the same clientId and sequenceCounter as in step 6 with value DataValue2	**[RUN<RUN_Results2>]** runnable has been started by an AsynchronousServerCallReturnsEvent
Step 8	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Read	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK with value DataValue2
Post-conditions	None	

3.3.13 [ATS_RTE_00065] Test inter/intra-ECU C-S with operations of 2 ports mapped on one runnable

Test Objective	Test inter/intra-ECU C-S with operations of 2 ports mapped on one runnable		
ID	ATS_RTE_00065	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed

Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001	
Trace to SWS Item	RTE: SWS_Rte_08002	
Requirements / Reference to Test Environment	UC01.02.02	
Configuration Parameters	See UC01.02.02 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Write1A) Tester_Client_2 * port Client2B * runnable RUN_Client2 (ascp_Write2B) Tester_Server * port ServerA * port ServerB * runnable serverRead (invoked by the Write operation on both ports) frames WRITE_A_REQ, WRITE_A_RES for the inter-ECU communication to ServerA (operation Write) frames WRITE_B_REQ, WRITE_B_RES for the inter-ECU communication to ServerB (operation Write) Client1A connected to ServerA Client2B connected to ServerB The server/system shall be implemented/configured in such way that no requests are fully processed before all requests are issued.	
Summary	The goal of this test is to check the behavior of Client-Server communication when 2 operations are mapped on the same runnable entity (on server side) and clients are both local and remote. 2 server ports are mapped to the same server runnable, and are connected to remote and local clients. The servers are configured with a ServerComSpec.queueLength = 2. The test manager sends requests on the bus, and checks, on the bus, that the first 2 requests are handled correctly and the subsequent requests results in a RTE_E_LIMIT error.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]**	

	start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[LT]** send frame WRITE_B_REQ	
Step 4	**[LT]** send frame WRITE_A_REQ	
Step 6	**[CP]** WAIT 100ms	**[LT]** frame WRITE_B_RES has been transmitted with clientId and sequenceCounter from step 3, and return value RTE_E_OK frame WRITE_A_RES has been transmitted with clientId and sequenceCounter from step 4, and return value RTE_E_LIMIT
Step 7	-	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 8	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_Write	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK
Step 9	**[CP]** terminate RUN_Client2 and RUN_Results2	
Post-conditions	None	

3.3.14 [ATS_RTE_00071] Test n:1 inter-ECU Client-Server communication (remote clients)

Test Objective	Test n:1 inter-ECU Client-Server communication (remote clients)		
ID	ATS_RTE_00071	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item			
Requirements / Reference to Test Environment	UC01.02.02		

Configuration Parameters	See UC01.02.02 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Server * port RepeatS * runnable serverRepeat, which repeats its IN parameter into its OUT parameter frames REPEAT_REQ, REPEAT_RES In addition: For ServerRepeat: ServerComSpec.queueLength >= 3	
Summary	The goal of this test is to check the behavior of n:1 inter-ECU when a server is invoked by remote clients. The server is located on the SUT. It repeats in its OUT parameter the value received in its IN parameter. 3 requests (with 3 different IN parameter values) are sent by 2 different clients (clientId1, clientId2). The test manager checks that each request receives the correct response (value is repeated)	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT]** send frames: • REPEAT_REQ with clientId1, sequenceCounter=1, IN parameter=DataValue1 • REPEAT_REQ with clientId2, sequenceCounter=1, IN parameter=DataValue2 • REPEAT_REQ with clientId1, sequenceCounter=2, IN parameter=DataValue3	•
Step 2	**[CP]** WAIT 100ms	**[RUN<serverRepeat>]** runnable has been started 3 times, with IN parameter successively DataValue1, DataValue2, DataValue3
Step 3		**[LT]** The following frames have been transmitted by SUT on <Bus>: • REPEAT_RES with clientId1, sequenceCounter=1, OUT parameter=DataValue1, return=RTE_E_OK

		• REPEAT_RES with clientId2, sequenceCounter=1, OUT parameter=DataValue2, return=RTE_E_OK • REPEAT_RES with clientId1, sequenceCounter=2, OUT parameter=DataValue3, return=RTE_E_OK
Post-conditions	None	

3.3.15 [ATS_RTE_00141] Test inter-ECU Client-Server communication - array on client side

Test Objective	Test inter-ECU Client-Server communication - array on client side		
ID	ATS_RTE_00141	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_08761		
Requirements / Reference to Test Environment	UC01.02.01		
Configuration Parameters	See UC01.05 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_4 * port Repeat4 * runnable RUN_Client4 (sscp_RepeatArray4) frame REPEATARRAY_REQ, which include a clientId, a sequenceCounter and the 3 uint8 values of the array frame REPEATARRAY_RES, which include a clientId, a sequenceCounter, the 3 uint8 values of the array, and a return value The client serializer concatenate clientId, sequenceCounter, and the 3 array values as uint8 signals The client deserializer extracts in the byte stream: clientId, sequenceCounter, array[0], array[1], array[2], return value in this order (each value is a uint8)		
Summary	The goal of this test is to check the serialization (resp. deserialization) by a client of an array IN (resp. OUT) parameter in case of remote server invocation.		

	The client, on SUT, invokes a server with an array IN parameter. The test manager checks that the IN parameter is correctly serialized in uint8 signals in the transmitted frame. The test manager sends back the same data in a response frame and checks that the client receives the correct array values.
Needed Adaptation to other Releases	
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client4	
Step 2	**[RUN<RUN_Client4>]** ArrayIn[0]=0x55 ArrayIn[1]=0xAA ArrayIn[2]=0x55 execute Rte_Call_Repeat4_RepeatArray(&ArrayIn[0], &ArrayOut[0])	**[LT]** frame REPEATARRAY_REQ is transmitted with byte stream: [clientId,sequenceCounter,0x55,0xAA,0x55]
Step 3	**[LT]** send frame REPEATARRAY_RES with byte stream: [clientId,sequenceCounter,0x55,0xAA,0x55,RTE_E_OK]	**[RUN<RUN_Client4>]** Rte_Call returns RTE_E_OK and provides the following array: ArrayOut[0]=0x55 ArrayOut[1]=0xAA ArrayOut[2]=0x55
Post-conditions	None	

3.3.16 [ATS_RTE_00142] Test inter-ECU Client-Server communication - array on server side

Test Objective	Test inter-ECU Client-Server communication - array on server side		
ID	ATS_RTE_00142	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance	ATR: ATR_ATR_00002		

Test Document		
Trace to SWS Item	RTE: SWS_Rte_08762	
Requirements / Reference to Test Environment	UC01.02.02	
Configuration Parameters	See UC01.02.02 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Server * port RepeatS * runnable repeatArray frame REPEATARRAY_REQ, which include a clientId, a sequenceCounter and the 3 uint8 values of the array frame REPEATARRAY_RES, which include a clientId, a sequenceCounter, the 3 uint8 values of the array, and a return value	
Summary	The goal of this test is to check the serialization (resp. deserialization) by a server of an array IN (resp. OUT) parameter in case of server invocation by a remote client. The test manager sends a frame to invoke the server. It checks that the server is executed and received the correct IN parameter, and then checks that the server's response is correctly transmitted on the bus.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT]** send frame REPEATARRAY_REQ with byte stream: [clientId,sequenceCounter,0x55,0xAA,0x55]	
Step 2	**[CP]** WAIT 100ms	**[RUN<serverRepeatArray>]** runnable has been started by OperationInvokedEvent and received IN parameter [0xAA,0x55,0xAA]
Step 3	**[RUN<serverRepeatArray>]** copy IN parameter into OUT parameter and return RTE_E_OK	**[LT]** frame REPEATARRAY_RES is transmitted with byte stream: [clientId,sequenceCounter,0xAA,0x55,0xAA,RTE_E_OK]
Post-conditions	None	

3.3.17 [ATS_RTE_00206] Test intra-ECU Client-Server communication - independence of operations

Test Objective	Test intra-ECU Client-Server communication - independence of operations		
ID	ATS_RTE_00206	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules		**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_04517 RTE: SWS_Rte_04518		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_Read1A) Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_Write2A) Tester_Server * port ServerA * runnable serverRead * runnable serverWrite Client1A and Client2A connected to ServerA.		
Summary	The goal of this test case is to check that operations of a server can be invoked independently. Tester_Client_1 invokes the Write operation (synchronous server call). Tester_Client_2 invokes the Read operation (asynchronous server call). The test manager check that both invocations succeed and give the appropriate result.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start RUN_Client1, RUN_Client2		

Step 2	**[RUN<RUN_Client2>]** execute Rte_Call_Client2A_Write(DataValue1)	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[RUN<RUN_Client1>]** execute Rte_Call_Client1A_Read	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK and data returned in DataValue1
Step 4	**[CP]** terminate RUN_Client1, RUN_Client2	
Post-conditions	None	

3.3.18 [ATS_RTE_00242] Test concurrent activation of server (runnable not mapped to task)

Test Objective	Test concurrent activation of server (runnable not mapped to task)		
ID	ATS_RTE_00242	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_03523		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Reentrant1 * runnable RUN_Client1 (sscp_Reentrant1) Tester_Client_3 * port Reentrant3 * runnable RUN_Client3 (sscp_Reentrant3) Tester_Server * port ReentrantS * runnable serverReentrant, canBeInvokedConcurrently=true Reentrant1 and Reentrant3 connected to ReentrantS. serverReentrant is not mapped to a task The serverReentrant/system shall be implemented/configured in such way that the request of Tester_Client_1 is being processed when the request of Tester_Client_3 is issued.		

Summary	The goal of this test case is to check the support of the RTE for servers that canBeInvokedConcurrently when they are not mapped to a task (i.e. direct function call). In this test case, the Tester_Client_1 and Tester_Client_3 will call simultaneously the server operation Reentrant (the request from Tester_Client_3 is issued while the Tester_Client_1 request is processed). The test manager checks that the requests of Tester_Client_1 and Tester_Client_3 are handled correctly. HINT: The server could for example trigger the execution of RUN_Client3 depending on its parameter, and RUN_Client3 can be mapped to a task with an higher priority or the server could change its duration based on the its parameter.	
Needed Adaptation to other Releases		
Pre-conditions		
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client1	
Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Reentrant1_Reentrant(0x55)	
Step 3	**[CP]** start RUN_Client3 before the end of the server execution	
Step 4	**[RUN<RUN_Client3>]** execute Rte_Call_Reentrant3_Reentrant(0xAA)	**[RUN<serverReentrant>]** runnable has been started twice with parameters 0x55 and 0xAA. 2 instances of the runnable have been executed simultaneously.
Step 5		**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK
Step 6		**[RUN<RUN_Client3>]** Rte_Call returns RTE_E_OK
Post-conditions		

3.3.19 [ATS_RTE_00249] Test concurrent activation of server (runnable mapped to task)

Test Objective	Test concurrent activation of server (runnable mapped to task)		
ID	ATS_RTE_00249	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2

Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item			
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Reentrant1 * runnable RUN_Client1 (sscp_Reentrant1) Tester_Client_3 * port Reentrant3 * runnable RUN_Client3 (sscp_Reentrant3) Tester_Server * port ReentrantS * runnable serverReentrant, canBeInvokedConcurrently=true Reentrant1 and Reentrant3 connected to ReentrantS. serverReentrant is mapped to a task The serverReentrant/system shall be implemented/configured in such way that the request of Tester_Client_1 is being processed when the request of Tester_Client_3 is issued.		
Summary	The goal of this test case is to check the support of the RTE for servers that canBeInvokedConcurrently when they are mapped to a task. In this test case, the Tester_Client_1 and Tester_Client_3 will call simultaneously the server operation Reentrant (the request from Tester_Client_3 is issued while the Tester_Client_1 request is processed). The test manager checks that the requests of Tester_Client_1 and Tester_Client_3 are handled correctly. HINT: The server could for example, depending on its parameter, trigger the execution of RUN_Client3 mapped to a task with an higher priority or the server could change its duration based on the its parameter to let the TCP trigger RUN_Client3.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start RUN_Client1		

Step 2	**[RUN<RUN_Client1>]** execute Rte_Call_Reentrant1_Reentrant(0x55)	
Step 3	**[CP]** start RUN_Client3 before the end of the server execution	
Step 4	**[RUN<RUN_Client3>]** execute Rte_Call_Reentrant3_Reentrant(0xAA)	**[RUN<serverReentrant>]** runnable has been started twice with parameters 0x55 and 0xAA.
Step 5		**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK
Step 6		**[RUN<RUN_Client3>]** Rte_Call returns RTE_E_OK
Post-conditions	None	

3.3.20 [ATS_RTE_00846] Test synchronous server call for 1:1 intra-ECU Client-Server communication for INOUT Argument

Test Objective	Test synchronous server call for 1:1 intra-ECU Client-Server communication for INOUT Argument		
ID	ATS_RTE_00846	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00002		
Trace to SWS Item	RTE: SWS_Rte_01020 RTE: SWS_Rte_01102 RTE: SWS_Rte_01104 RTE: SWS_Rte_01166 RTE: SWS_Rte_01293		
Requirements / Reference to Test Environment	UC01.01		
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_1 * port Client1A * runnable RUN_Client1 (sscp_ReadWrite1A) Tester_Server * port ServerA * runnable serverReadWrite		

	Client1A connected to ServerA. ServerA triggers serverReadWrite.
Summary	The goal of this test is to check the behavior of synchronous server call in case of 1:1 Intra-ECU Client-Server communication for INOUT argument. In this testcase it is checked if the server receives the input value provided by the client and also the server updates the new data value to the client as part of INOUT parameter. The Test Manager checks that the operations are handled correctly.
Needed Adaptation to other Releases	None
Pre-conditions	None
Main Test Execution	

Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client1	
Step 2	**[RUN<RUN_Client1>]** set data to 0x55 and execute Rte_Call_Client1A_ReadWrite(&data)	**[RUN<RUN_Client1>]** Rte_Call returns RTE_E_OK, data returned is 0xAA
Step 3	**[CP]** terminate RUN_Client1	
Post-conditions	None	

3.3.21 [ATS_RTE_00852] Test asynchronous server call for 1:1 intra-ECU Client-Server communication for INOUT Argument

Test Objective	Test asynchronous server call for 1:1 intra-ECU Client-Server communication for INOUT Argument		
ID	ATS_RTE_00852	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00001		
Trace to SWS Item	RTE: SWS_Rte_01020 RTE: SWS_Rte_01102 RTE: SWS_Rte_01104 RTE: SWS_Rte_01111 RTE: SWS_Rte_01112 RTE: SWS_Rte_01166 RTE: SWS_Rte_01294 RTE: SWS_Rte_01296		

Requirements / Reference to Test Environment	UC01.01	
Configuration Parameters	See UC01.01 in chapter 3.1.2 Test Configuration. This test case uses: Tester_Client_2 * port Client2A * runnable RUN_Client2 (ascp_ReadWrite2A) * runnable RUN_Results2 (ascrp_ReadWrite2A, ASCRE_ReadWrite2A) Tester_Server * port ServerA * runnable serverReadWrite Client2A connected to ServerA. ServerA triggers serverReadWrite.	
Summary	The goal of this test is to check the behavior of asynchronous server calls in case of 1:1 intra-ECU Client-Server communication for an operation which has INOUT argument This test involves communication between Tester_Client_2 and Tester_Server. The Tester_Client_2 SW-C calls the local server for the ReadWrite operation. In this testcase it is checked if the server receives the input value provided by the client and also the server updates the new data value to the client as part of INOUT parameter of the Rte_Result API. The Test Manager checks that asynchronous server calls for the ReadWrite operation succeed (data is written and read correctly) and that an AsynchronousServerCallReturnsEvent is triggered to start a runnable of the client.	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start RUN_Client2	
Step 2	**[RUN<RUN_Client2>]** set data to 0x55 and execute Rte_Call_Client2A_ReadWrite(data)	**[RUN<RUN_Client2>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** WAIT 100ms	**[RUN<RUN_Results2>]** runnable has been started by AsynchronousServerCallReturnsEvent
Step 4	**[RUN<RUN_Results2>]** execute Rte_Result_Client2A_ReadWrite(&data)	**[RUN<RUN_Results2>]** Rte_Result returns RTE_E_OK, data returned is 0xAA

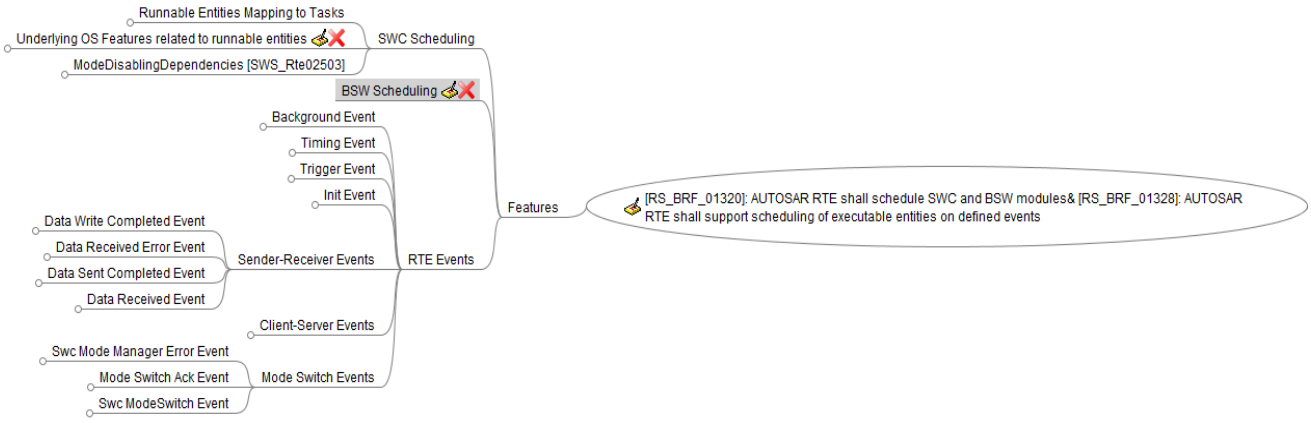
Post-conditions	None

4 RS_BRF_01320 & RS_BRF_01328 – Rte SWC scheduling and activation from events

4.1 General Test Objective and Approach

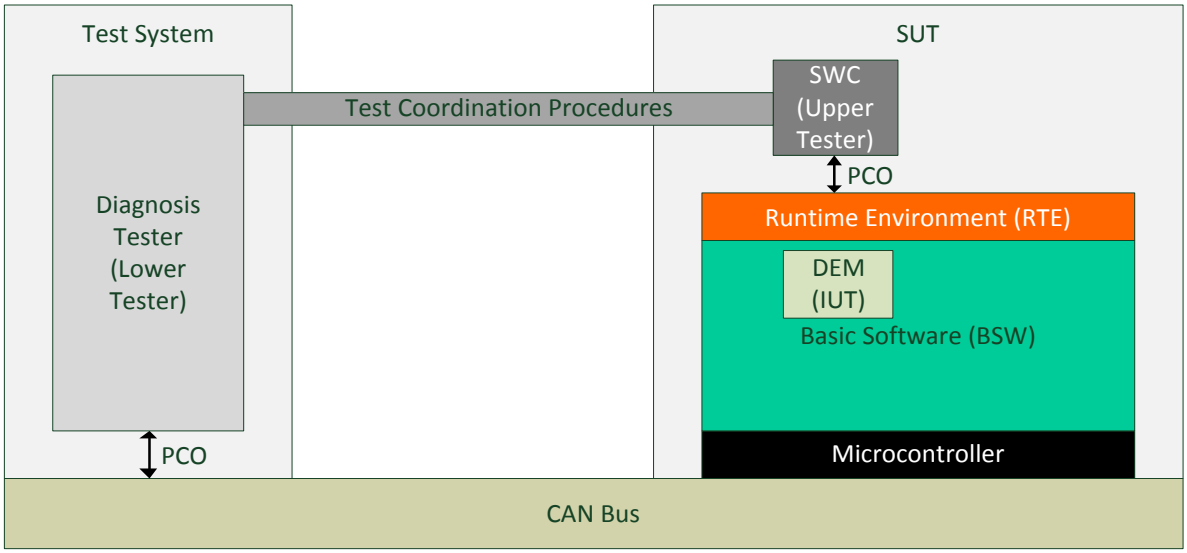
This test suite intends to cover the RTE SWC scheduling and activation from events features. As Acceptance Tests intends to cover ICC1 scope, this test suite has limited scope of tests cases (**not focusing on the underlying OS features**).

The contents of these features are detailed below:



4.1.1 Test System

4.1.1.1 Overview on Architecture



The test system architecture consists of one test bench and a System Under Test.

4.1.1.2 Specific Requirements

none

4.1.2 Test Configuration

This section describes sets of requirements on configuration.
These sets are later referenced by test cases.
No configuration files are provided. They need to be developed when the test suite is implemented.

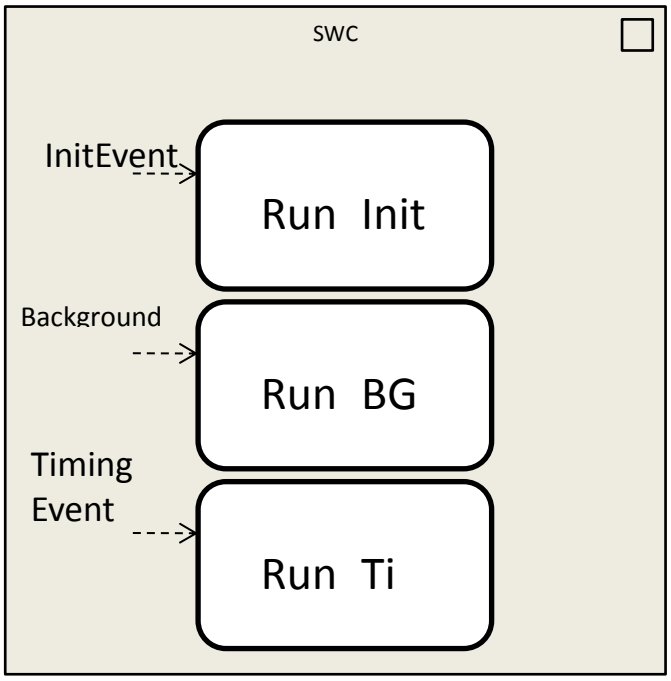
4.1.2.1 Required ECU Extract of System Description Files

In order to cover the required features, the requirements for configuration have been split in uses cases:

4.1.2.1.1 Use case 02.01: General Events activated runnables

This use case consists in verifying the 3 general events behavior:

Use Case 1 :
Runnables activated by General Events



S

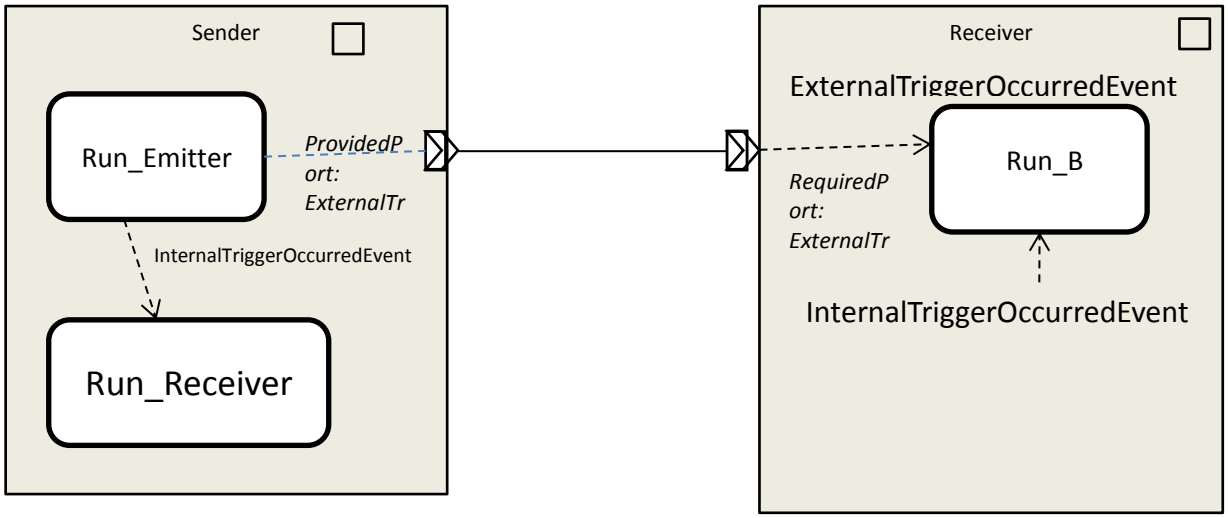
No other additional requirement for ECU Extract required.

4.1.2.1.2 Use case 02.02: Runnables activated by Trigger Events

This use case consists in applications which use triggers:

- InternalTriggerOccurredEvent
- ExternalTriggerOccurredEvent
- Queuing of Triggers

Use Case 2 :
Runnables activated by TriggerEvents



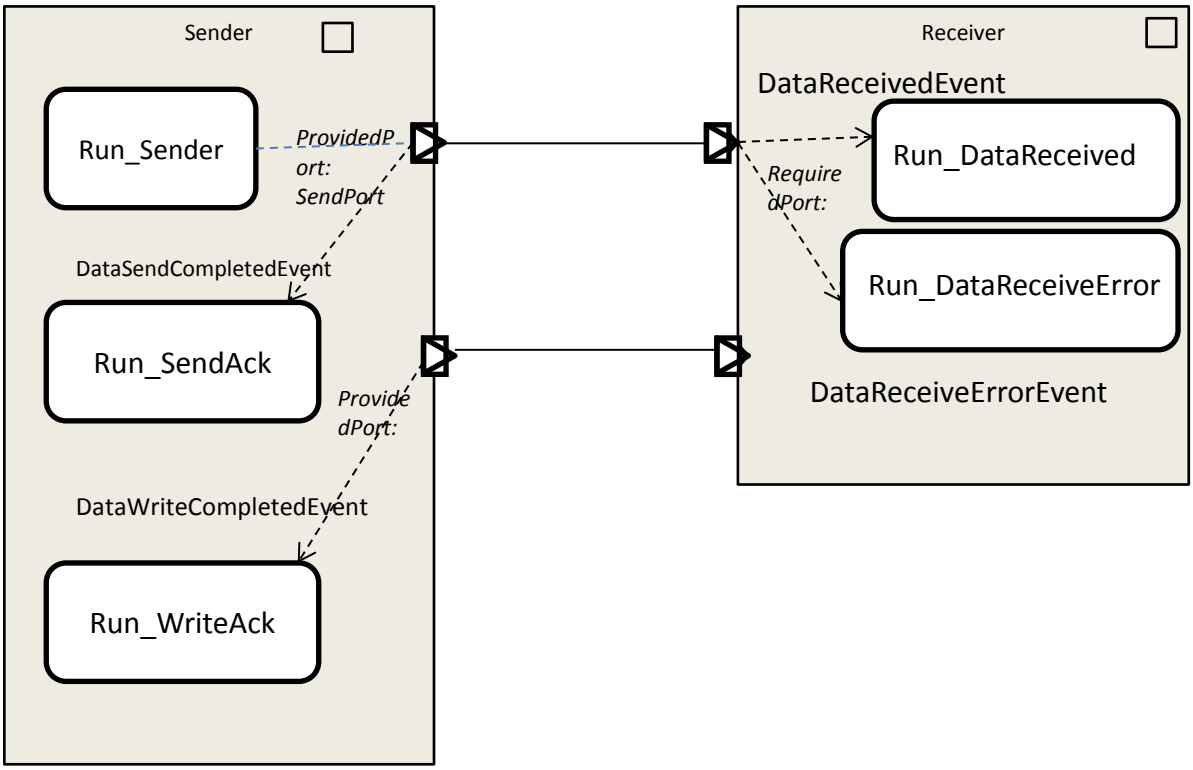
This test case uses 2 SWC located on SUT.

No external CAN frame defined.

Hint: RteTriggerSourceQueueLength shall be configured >= 2 for the ExternalTrigger

4.1.2.1.3 Use case 02.03: Runnables activated by Sender-Receiver related events

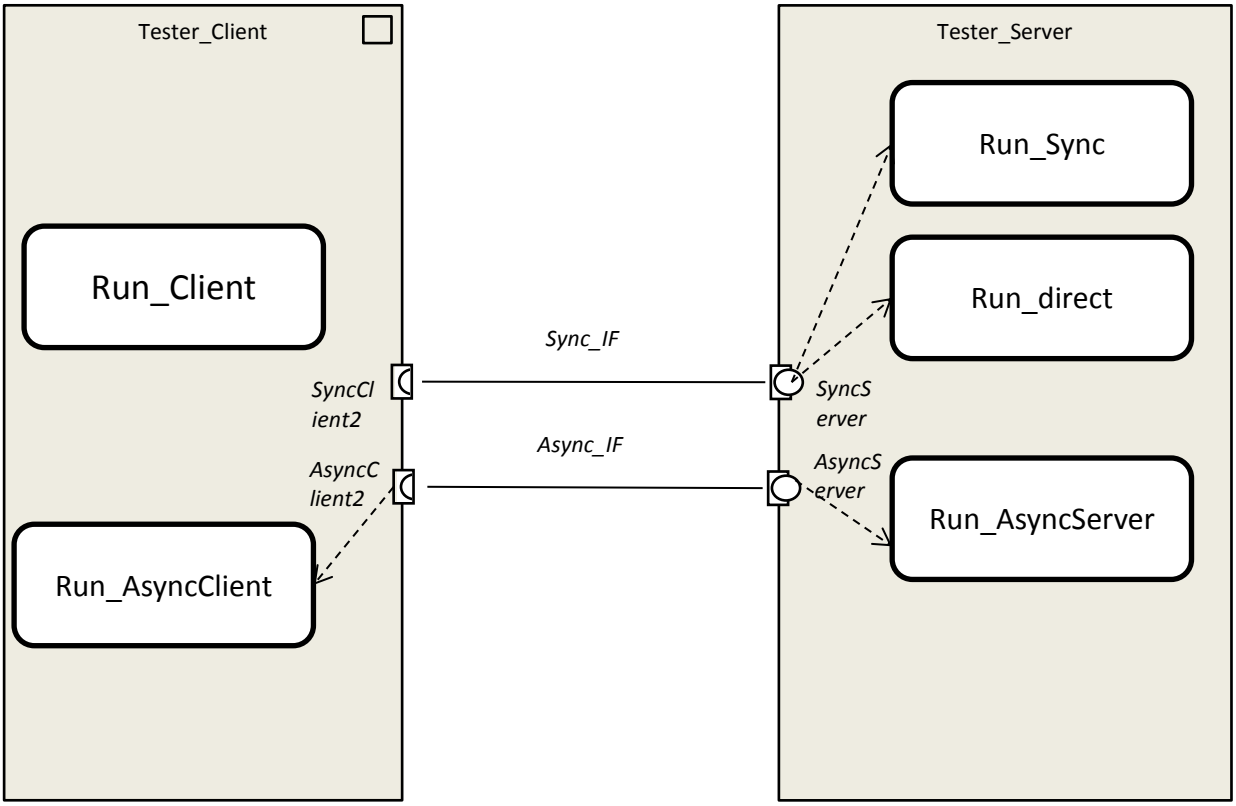
Use Case 3 :
Runnables activated by S/R Events



This use case intends to cover Sender-Receiver related events.

4.1.2.1.4 Use case 02.04: Runnables activated by Client-Server related events

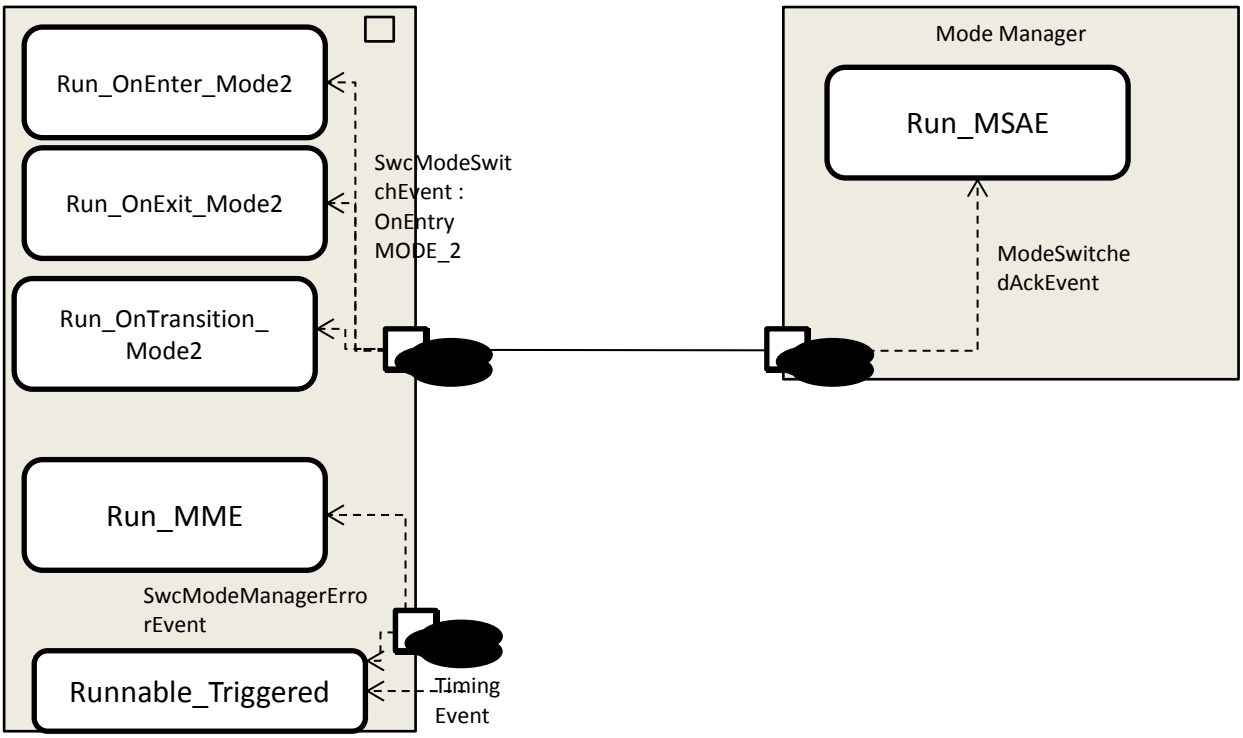
Use Case 4 :
Runnables activated by C/S Events



This use case intends to cover Client-Server related Events.

4.1.2.1.5 Use case 02.05: Runnables activated by Mode Switch related events

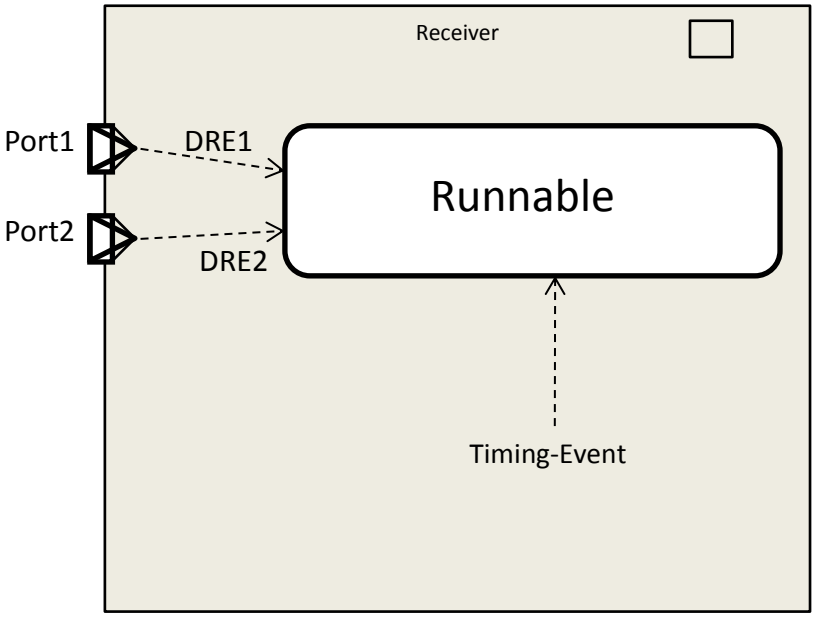
This use case intends to cover the ModeSwitch Activated runnables:



This Test case will use :

- ModeDeclarationGroup:
  - MODE_1 (Init Value)
  - MODE_2
  - MODE_3
- One SwcModeManager:
  - Run_MSAE is activated consecutively to a ModeSwitchedAckEvent
- OneSwcModeUser:
  - Run_OnEnter_Mode2 is activated when entering in MODE_2
  - Run_OnExit_Mode2 is activated when exiting MODE_2
  - Run_OnTransition is activated when transition to MODE_2
  - Runnable_Triggered:
    - Activated by TiminEvent 100ms
    - ModeDisablingDependency when entering in MODE_3

4.1.2.1.6 Use case 02.06: Runnable activated by Multiple Events



This use case intends to cover runnable activation by multiple events and check the activation reason at each runnable activation.

The Runnable will check the ActivationReason (DRE1, DRE2 or Timing-Event).

To avoid ICC3 parameters:

- Use a SenderRunnable connected to Port1 & Port2
- Sender SWC and Receiver runsInsideExclusiveArea EA1

4.1.2.2 Required ECU Configuration Description Files

There are no generic requirements on ECU Configuration files. Individual test cases may have specific requirements on the RTE configuration.

4.1.2.3 Required Software Component Description Files

Requirements on on Software Component Description files are provided in the section 4.1.2.1 Required ECU Extract of System Description Files, together with the connections of the different components.

4.1.2.4 Mandatory vs. Customizable Parts

Not Applicable

4.1.3 Test Case Design

Not Applicable

4.2 Re-usable Test Steps

Not Applicable

4.3 Test Cases

4.3.1 [ATS_RTE_00116] Test Behavior of runnables activated by "General" Events

Test Objective	Test Behavior of runnables activated by "General" Events		
ID	ATS_RTE_00116	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE, OS	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022		
Trace to SWS Item	RTE: SWS_Rte_01126 RTE: SWS_Rte_01130 RTE: SWS_Rte_01131 RTE: SWS_Rte_01132 RTE: SWS_Rte_02202 RTE: SWS_Rte_06728 RTE: SWS_Rte_06748 RTE: SWS_Rte_06761 RTE: SWS_Rte_07177		
Requirements / Reference to Test Environment	Use case 02.01: Runnables activated on general events		
Configuration Parameters	1 Runnable activated by BackgroundEvent (Background Task) 1 Runnable with TimingEvent Runnable (100ms) 1 Runnable with InitEvent Hint: The test case relies on a background task to be activated. The test environment has to ensure that such task can be activated (ECU is not kept busy for other processes not related to this test case)		
Summary	This test case checks that : InitEvent Runnable is activated after initialization TimingEvent is activated every 100ms (timing measurement) BackgroundEvent runnable is activated when the ECU is not busy (in the background task)		
Needed Adaptation to other Releases	**Needed Adaptation for Release [4.0.3]** Configuration: [low] — InitEvents are not available in R4.0. Test Steps: [low] — The associated test steps shall be removed		
Pre-conditions	ECU powered off		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	**[CP]** Power On ECU WAIT 50ms	**[SWC]** InitEvent associted runnable has been activated after initialization
Step 2	**[SWC]** WAIT all RTE tasks to be released (return from all runnables, included test framework SWCs)	**[SWC]** BackgroundEvent associated runnable has been activated (note: future activation can be ignored)
Step 3	**[CP]** WAIT 125ms	**[SWC]** TimingEvent associated runnable has been activated.
Step 4	**[SWC]** save TimeStamp1=GetTimeStamp() when TimingEvent associated runnable is activated	
Step 5	**[CP]** WAIT 125ms	**[SWC]** TimingEvent associated runnable has been activated.
Step 6	**[SWC]** save TimeStamp2=GetTimeStamp() when TimingEvent associated runnable is activated	**[SWC]** TimeStamp2-TimeStamp1 == 100ms (+/- 1ms)
Post-conditions	none	

4.3.2 [ATS_RTE_00117] Test Runnables activated by Trigger Events

Test Objective	Test Runnables activated by Trigger Events		
ID	ATS_RTE_00117	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, OS	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022		
Trace to SWS Item	RTE: SWS_Rte_07087 RTE: SWS_Rte_07090 RTE: SWS_Rte_07207 RTE: SWS_Rte_07208 RTE: SWS_Rte_07212 RTE: SWS_Rte_07213 RTE: SWS_Rte_07543		
Requirements / Reference to Test Environment	Use case 02.02 : Runnables activated by Trigger Event		

Configuration Parameters	1 SWC with 2 runnables : 1 ExternalTrigger Provided Port interface A runnable Run_Emitter which is identified with : one InternalTriggeringPoint InternalTrigger one ExternalTriggeringPoint ExternalTrigger A runnable Run_Receiver which is activated by an InternalTriggerOccurredEvent referencing InternalTrigger 1 SWC with 1 runnable, Run_B, activated by ExternalTriggerOccurredEvent 1 ExternalTrigger RequiredPort interface, connected to the first SWC 1 ExternalTrigger Provided Port interface with queued swImplPolicy Hint: RteTriggerSourceQueueLength shall be configured >= 2 for the ExternalTrigger	
Summary	This test case intends to check the behavior of Runnables activated by: - ExternalTriggerOccurredEvent - InternalTriggerOccurredEvent This test will also cover the queuing of triggers.	
Needed Adaptation to other Releases		
Pre-conditions	ECU powered off	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Power ON ECU	
Step 2	**[CP]** Start Run_Emitter	
Step 3	**[RUN<Run_Emitter>]** Call Rte_Trigger() for ExternalTriggerOccurredEvent.	**[RUN<Run_B>]** Run_B has been activated
Step 4	**[RUN<Run_Emitter>]** Trig InternalTrigger by calling Rte_IrTrigger()	**[RUN<Run_Receiver>]** Run_Receiver has been activated
Step 5	**[RUN<Run_Emitter>]** Trig twice ExternalTrigger by calling twice Rte_Trigger()	**[RUN<Run_B>]** Run_B has been activated twice
Post-conditions	None	

4.3.3 [ATS_RTE_00118] Test Runnables activated from S/R communication

Test Objective	Test Runnables activated from S/R communication

ID	ATS_RTE_00118	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022		
Trace to SWS Item	RTE: SWS_Rte_01135 RTE: SWS_Rte_01137 RTE: SWS_Rte_01359		
Requirements / Reference to Test Environment	Use case 02.03 : Runnables activated by S/R communication		
Configuration Parameters	1 SWC Sender: 1 Sender Port interface 1 runnable Run_Sender 1 runnable Run_SendAck activated by DataSentCompletedEvent 1 runnable Run_WriteAck activated by DataWriteCompletedEvent 1 SWC Receiver: 1 Receiver Port Interface - ReceiverPort 1 runnable Run_DataReceived activated by DataReceivedEvent on ReceiverPort 1 runnable Run_Data_ReceiveError activated by DataReceiveErrorEvent on ExternalReceiverPort ExternalReceiverPort dataelement mapped to an incoming BUS signal Hint: It is expected that the system is configured such that * Run_DataReceived, Run_SendAck, Run_WriteAck will be executed less than 1ms after their activation (which is triggered from Run_Sender) * Run_ReceiveError will be executed less than 1ms after reception of an invalid signal from the bus		
Summary	The goal of this test case is to cover the runnable activations following S/R communication. The following Events are checked: - DataReceivedEvent Runnable activation - DataWriteCompletedEvent Runnable activation - DataReceiveErrorEvent Runnable activation - DataSendCompletedEvent Runnable activation The test generates each event kind and then checks if the runnable has been activated		
Needed Adaptation to other Releases			
Pre-conditions	ECU Powered off		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Power ON ECU		
Step 2	**[CP]**		

	Start Run_Sender	
Step 3	**[RUN<Run_Sender>]** Send Data on SenderPort interface by calling Rte_Send() WAIT 1ms	**[RUN<Run_DataReceived>]** Run_DataReceived has been activated by DataReceivedEvent
Step 4		**[RUN<Run_SendAck>]** Run_SendAck has been activated by DataSendCompletedEvent
Step 5	**[CP]** WAIT 1ms	
Step 6	**[RUN<Run_Sender>]** Write data on WritePort interface (Rte_IWrite()) WAIT 1ms	**[RUN<Run_WriteAck>]** Run_WriteAck has been activated by DataWriteCompletedEvent
Step 7	**[CP]** WAIT 1ms	
Step 8	**[LT]** Send Data on Bus corresponding to dataElement in ExtSenderPort with invalid value	
Step 9	**[CP]** WAIT 100ms	**[RUN<Run_ReceiveError>]** Run_ReceiveError has been activated by DataReceiveErrorEvent
Post-conditions	none	

4.3.4 [ATS_RTE_00119] Test Runnables activated from C/S communication

Test Objective	Test Runnables activated from C/S communication		
ID	ATS_RTE_00119	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022		
Trace to SWS Item			
Requirements / Reference to Test Environment	Use case 02.04: Runnables activated by C/S		

Configuration Parameters	1 SWC Client 1 runnable Run_Client with 2 SynchronousServerCallPoints (operation1 and operation2) and one AsynchronousServerCallPoint (operation3) 1 runnable Run_AsyncClient activated by an AsynchronousServerCallReturnsEvent 1 SWC Server 1 runnable Run_Sync with attribute CanBeInvokedConcurrently set to FALSE activated by OperationInvokedEvent (operation1) 1 runnable Run_direct with attribute CanBeInvokedConcurrently set to TRUE activated by an OperationInvokedEvent (operation2) 1 runnable run_asyncServer activated by OperationInvokedEvent (operation3)	
Summary	The following activations will be tested: - 1 Server Runnable with canBeInvokedConcurrently=FALSE activated by an OperationInvokedEvent triggered by a synchronous client - 1 Server Runnable with canBeInvokedConcurrently=TRUE activated by an OperationInvokedEvent triggered by a synchronous client - 1 Server Runnable activated by an OperationInvokedEvent triggered by an asynchronous client - 1 Runnable activated by AsynchronousServerCallReturnsEvent The test will activate each event and check if the server/client are called consecutively.	
Needed Adaptation to other Releases		
Pre-conditions	ECU Powered off	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Power_ON ECU	
Step 2	**[CP]** Start RUN_Client	
Step 3	**[RUN<Run_Client>]** Call operation operation1()	
Step 4	**[CP]** WAIT 1ms	**[RUN<Run_sync>]** Run_Sync has been activated by the OperationInvokedEvent
Step 5	**[RUN<Run_Client>]** Call operation operation2()	**[RUN<Run_direct>]** Run_direct has been immediately activated
Step 6	**[RUN<Run_Client>]** Call operation operation3()	
Step 7	**[CP]** WAIT 1ms	**[RUN<Run_asyncServer>]** Run_asyncServer has been activated by OperationInvokedEvent

Step 8		**[RUN<Run_AsyncClient>]** Run_AsyncClient has been activated by AsynchronousServerCallReturnsEvent
Post-conditions	none	

4.3.5 [ATS_RTE_00121] Test Runnable activated by Mode Switchs

Test Objective	Test Runnable activated by Mode Switchs		
ID	ATS_RTE_00121	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022		
Trace to SWS Item	RTE: SWS_Rte_02512 RTE: SWS_Rte_02758 RTE: SWS_Rte_06771		
Requirements / Reference to Test Environment	Use Case 02.05 : Mode Switch related events		
Configuration Parameters	A mode switch interface {MODE_1, MODE_2, MODE_3} with initial mode = MODE_1 An atomic software component with a required port prototype typed by this interface. A runnable activated on mode transition 'SwcModeSwitchEvent' (on transition from MODE_1 to MODE_2), A runnable activated on mode transition 'SwcModeSwitchEvent' (on enter in MODE_2), A runnable activated on mode transition 'SwcModeSwitchEvent' (on exit MODE_2), A runnable triggered on TimingEvent but disabled in MODE_3 with a ModeDisablingDependency An atomic software component with a provider port prototype typed by this interface. A runnable activated on ModeSwitchedAckEvent A runnable activated on SwcModeManagerErrorEvent		
Summary	The goal of this test case is to check the correct activation of runnables after the following events: - SwcModeManagerErrorEvent - SwcModeSwitchEvent - ModeSwitchedAckEvent - ModeDisablindDependency A SWC will request a Mode Change through a ModeSwitch interface, then the Mode Manager will generate a SwcModeSwitchEvent. Check the activation of the runnables by the SwcModeSwitchEvent and the ModeSwitchedAckEvent.		

	Then restart the operation, generate a SwcModeManagerErrorEvent and check activation of the runnable. Hint: to check the activation of a runnable, the runnable can wrap a counter increased at each activation and publised through a sender/receiver interface.
Needed Adaptation to other Releases	**Needed Adaptation for Release [4.0.3]** Configuration: [Low] — ModeManagerErrorEvent is not available. Test Steps: [Low] — The associated test steps shall be removed
Pre-conditions	ECU Powered off

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Power ON ECU	
Step 2	**[SWC]** Call ModeSwitch interface to request mode switch to MODE_2	**[SWC]** Runnable activated on transition from MODE_1 to MODE_2 has been called Runnable activated on entering MODE_2 has been called Runnable activated on ModeSwitchedAckEvent has been called
Step 3	**[SWC]** Call ModeSwitch interfacte to request mode change to MODE_3	**[SWC]** Runnable activated on exiting MODE_2 has been called
Step 4	**[CP]** Wait 10 times the period of the TimingEvent	**[SWC]** The runnable activated on TimingEvent has not been called during that time
Step 5	**[CP]** Call ModeSwitch Interface with an UnknownMode (not MODE_1, MODE_2 or MODE_3 values)	**[SWC]** Runnable activated on SwcModeManagerErrorEvent has been called
Post-conditions	none	

4.3.6 [ATS_RTE_00132] Test Runnable activated by Multiple Events

Test Objective	Test Runnable activated by Multiple Events		
ID	ATS_RTE_00132	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00022
Trace to SWS Item	RTE: SWS_Rte_01126 RTE: SWS_Rte_03520 RTE: SWS_Rte_03524 RTE: SWS_Rte_08051 RTE: SWS_Rte_08054 RTE: SWS_Rte_08055 RTE: SWS_Rte_08060
Requirements / Reference to Test Environment	Use Case 02.06: Runnable activated by Multiple Events
Configuration Parameters	1 runnable in a SWC sender The SWC Sender is connected with 2 DataSendPoints on 2 different Ports. -> 1 senderRunnable which can enter ExclusiveArea (EA1) as receiverRunnable -> 1 dataSendPoint to port SenderPort1 -> 1 dataSendPoint to port SenderPort2 1 receiverRunnable activated with by 2 DataReceivedEvents (DRE1, DRE2) and a TimingEvent (100ms) The receiverRunnable runsInsideExclusiveArea EA1 The receiverRunnable entity is configured with 3 ExecutableEntityActivationReasons referenced to the 3 different events DRE1 occurs when receiving data on Port1 DRE2 occurs when receiving data on Port2
Summary	The goal of this test is to test the RTE feature of Runnable activation by multiple events. For this test, the runnable will be activated by both a TimingEvent and a DataReceivedEvent The test Manager will start the runnable. Checks first that Runnable is activated periodically at the period configured in TimingEvent (100ms).Then, Send data from Test Framework. The Test manager will collect at each task activation the ActivationReason (DRE1, DRE2 or TimingEvent)
Needed Adaptation to other Releases	**Needed Adaptation for Release [4.0.3]** Configuration: [low] — The Activation Reason feature is new in R4.1.1, this part of test shall be removed to be compatible with R4.0.3. Test Steps: [low]
Pre-conditions	ECU Powered off

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Power ON ECU	
Step 2	**[CP]**	**[RUN<receiverRunnable>]**

	wait for 3 times the TimingEvent period	receiverRunnable has been activated and Rte_ActivatingEvent corresponds to TimingEvent
Step 3	**[CP]** WAIT 100ms	**[RUN<receiverRunnable>]** receiverRunnable has been activated and Rte_ActivatingEvent corresponds to TimingEvent
Step 4	**[RUN<senderRunnable>]** Send Data on SenderPort1 (Rte_Send() API).	**[RUN<receiverRunnable>]** receiverRunnable has been activated and Rte_ActivatingEvent corresponds to DRE1
Step 5	**[RUN<senderRunnable>]** Send Data on SenderPort2 (Rte_Send() API)	**[RUN<receiverRunnable>]** receiverRunnable has been activated and Rte_ActivatingEvent corresponds to DRE2
Step 6	**[RUN<senderRunnable>]** Enter Exclusive Area EA1 Send Data on SenderPort1 (Rte_Send() API) Send Data on SenderPort2 (Rte_Send() API) Exit Exclusive Area EA1	**[RUN<receiverRunnable>]** receiverRunnable has been activated and Rte_ActivatingEvent corresponds to DRE1 and DRE2
Post-conditions	none	

4.3.7 [ATS_RTE_00694] Waking Up A Runnable From WaitPoint On Occurrence Of ModeSwitchedAckEvent

Test Objective	Waking Up A Runnable From WaitPoint On Occurrence Of ModeSwitchedAckEvent		
ID	ATS_RTE_00694	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_02628 RTE: SWS_Rte_02629 RTE: SWS_Rte_02631 RTE: SWS_Rte_02632 RTE: SWS_Rte_02660 RTE: SWS_Rte_02667 RTE: SWS_Rte_02674 RTE: SWS_Rte_02676 RTE: SWS_Rte_02725 RTE: SWS_Rte_02729		
Requirements / Reference			

to Test Environment	
Configuration Parameters	A ModeDeclarationGroup MDG_TC69 with * 2 ModeDeclarations: InitMode=0 StartMode=1 * initialMode=InitMode A SW-C with * A PPortPrototype which - is typed with a ModeSwitchInterface with a ModeDeclarationGroupPrototype with type=MDG_TC69 - has a ModeSwitchSenderComSpec with a ModeSwitchedAckRequest * An RPortPrototype which - is typed by the same ModeSwitchInterface - is connected with the previous PPortPrototype * RunnableEntity_TC69 to execute the test sequence - with a modeSwitchPoint which references the ModeDeclarationGroupPrototype of the PPortPrototype - with a modeAccessPoint which references the ModeDeclarationGroupPrototype of the RPortPrototype - with a WaitPoint which references a ModeSwitchedAckEvent that in turn references the ModeDeclarationGroupPrototype of the PPortPrototype
Summary	This test case verifies that whenever ModeSwitchedAckEvent occurs the RTE shall wake up a runnable from WaitPoint (blocking Rte_SwitchAck API).
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC69>] Invoke Rte_Mode.	[RUN<RunnableEntity_TC69>] Rte_Mode returns RTE_MODE_MDG_69_InitMode.
Step 2	[RUN<RunnableEntity_TC69>] Invoke Rte_Switch(RTE_MODE_MDG_TC69_StartMode).	[RUN<RunnableEntity_TC69>] Rte_Switch returns RTE_E_OK.
Step 3	[RUN<RunnableEntity_TC69>] Invoke Rte_SwitchAck.	[RUN<RunnableEntity_TC69>] Rte_SwitchAck returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

4.3.8 [ATS_RTE_00707] MinimumStartInterval For A Runnable Entity

Test Objective	MinimumStartInterval For A Runnable Entity		
ID	ATS_RTE_00707	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			

Trace to SWS Item	RTE: SWS_Rte_02697
Requirements / Reference to Test Environment	
Configuration Parameters	SW-C * PPortPrototype - VariableDataPrototype_TC103_0, primitive data type, swImplPolicy standard * PPortPrototype - VariableDataPrototype_TC103_1, primitive data type, swImplPolicy standard * RunnableEntity_T103_0 - dataWriteAccess referencing VariableDataPrototype_TC103_0 - started by TimingEvent (period=100ms) - minimumStartInterval=50ms * RunnableEntity_T103_1 - dataWriteAccess referencing VariableDataPrototype_TC103_1 - started by a TimingEvent (period=100ms) - minimumStartInterval=150ms Communication matrix * Signal_TC103_0 - mapped to VariableDataPrototype_TC103_0 - transmitted by the ECU * ISignalIPdu_TC103_0 - configured with an EventControlledTiming - Signal_TC103_0 mapped to ISignalIPdu_TC103_0, transferProperty triggered * Signal_TC103_1 - mapped to VariableDataPrototype_TC103_1 - transmitted by the ECU * ISignalIPdu_TC103_1 - configured with an EventControlledTiming - Signal_TC103_1 mapped to ISignalIPdu_TC103_1, with transferProp
Summary	In the below test case a runnable entity RunnableEntity_TC103_0 is configured with minimum start interval as 50ms and RunnableEntity_TC103_1 with a minimum start interval of 150ms. The 3 RunnableEntities are mapped to the same periodic task. The RunnableEntities send different dataElements which trigger the transmission of different frames used to monitor the periodicity of each RunnableEntity.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Observe time between consecutive frames for ISignalIPdu_TC103_0.	[LT] The time between consecutive frames for ISignalIPdu_TC103_0 is lower than 150ms.
Step 2	[LT] Observe time between consecutive frames for ISignalIPdu_TC103_1.	[LT] The time between consecutive frames for ISignalIPdu_TC103_1 is between 150ms and 250ms.
Post-conditions	NONE	

5 RS_BRF_01376 – Rte Data Conversion

5.1 General Test Objective and Approach

This document intends to cover the RTE Data Conversion feature (AUTOSAR Feature [RS_BRF_01376]).

The sub-features to cover are described below:

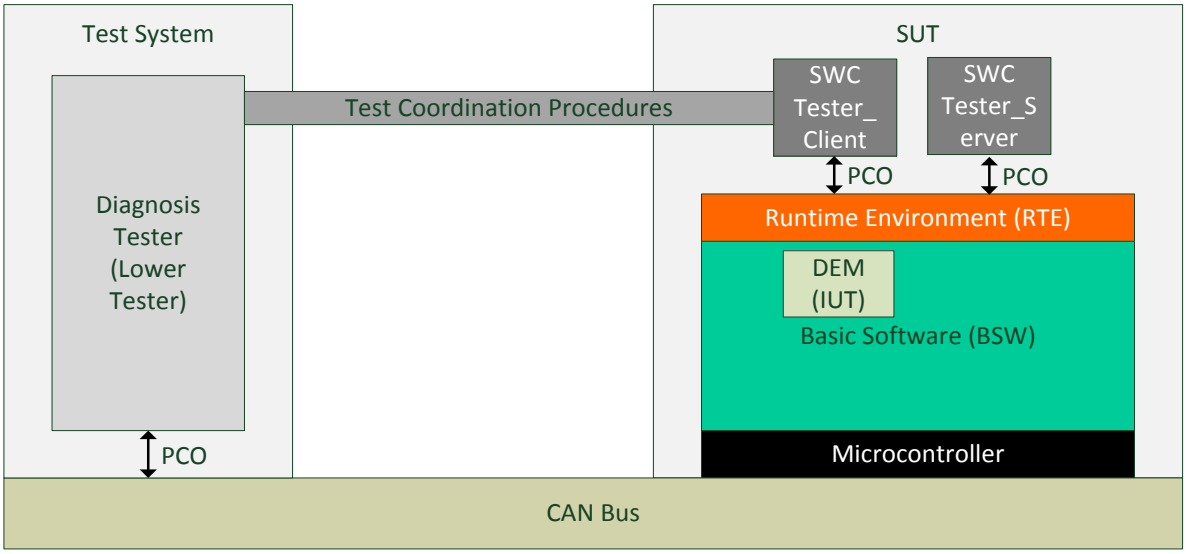
- ➔ DataTypes : RTE calculates numerical representation
- ➔ Port InterfaceElement Mapping and Data Conversion:
  - Port Interface Element Mapping:
    - Client-Server Interface Elements
    - Sender-Receiver Interface Elements
  - Data Conversions:
    - Linear Data Scaling
    - TextTable to TextTable
    - Mixed Linear scaled and TextTable to Mixed Linear scale and TextTable
    - Identical Conversion
    - Composite Representations
  - Network Representations
  - Range Checks During Runtime

These features intend to automatically convert by generated RTE a type 1 to a type 2.

5.1.1 Test System

5.1.1.1 Overview on Architecture

5.1.1.1.1 Use Case 03.01: Intra-ECU Data conversion for C/S interfaces

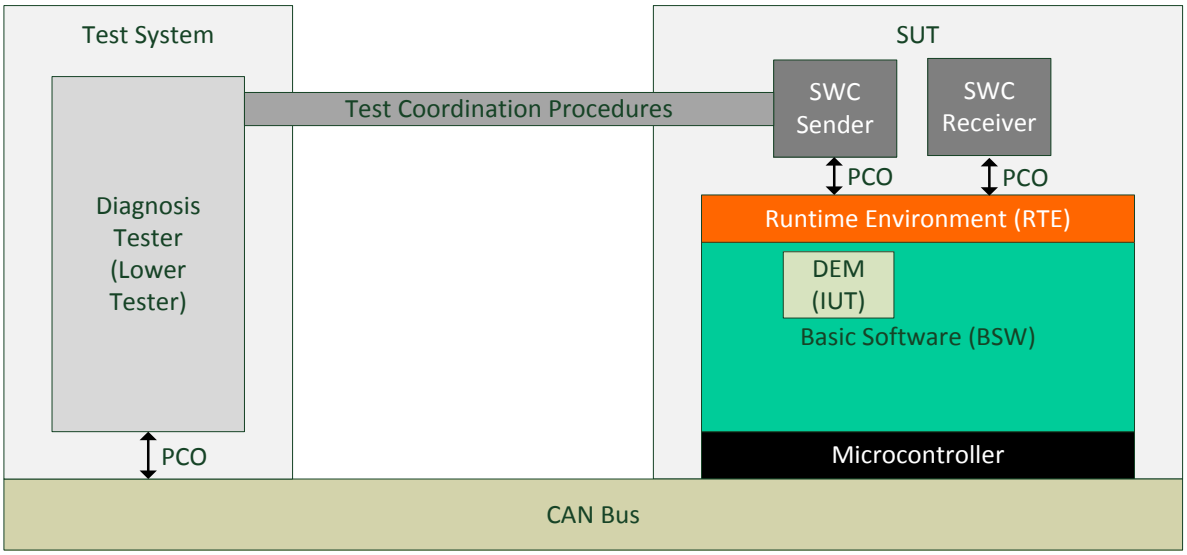


The test system architecture consists of 2 SWCs on the SUT:

- One SwC Tester_Client
- One SwC Tester_Server

Because the intention of these tests cases is to validate types, the used types and computation methods will be described in each test case.

5.1.1.1.2 Use Case 03.02: Intra-ECU Data conversion for S/R interfaces



The test system architecture consists of 2 SWCs on the SUT:

- One SwC Sender
- One SwC Receiver

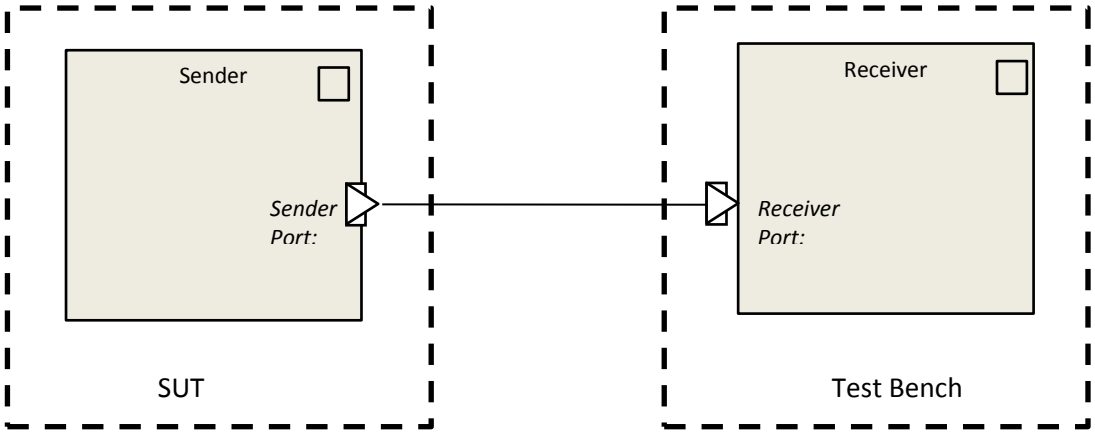
Because the intention of these tests cases is to validate types, the used types and computation methods will be described in each test case.

The intention of each test case will be to cover the automatic rescaling at port feature (TextTable to TextTable, Linear scale, …).

5.1.1.1.3 Use Case 03.03: Inter-ECU Network Representations

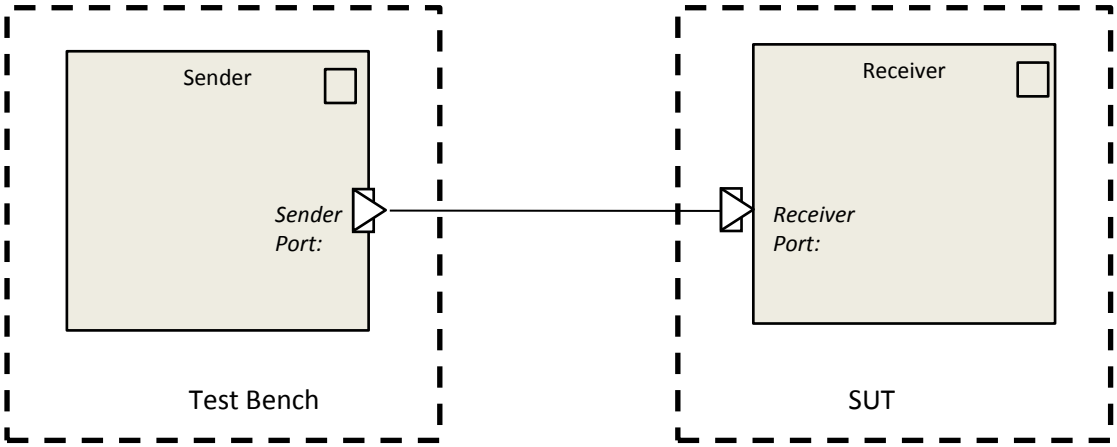
Use Case 3.1 : Sender on SUT

Use case 3.1: inter-ECU Network Representations on S/R Interface



Use case 3.2 : Sender on Test Bench

Use case 3.2: inter-ECU Network Representations on S/R Interface



The test system architecture consists of 2 SWCs:

- One SwC Sender
- One SwC Receiver

On Use case 3.1, the Sender is located on SUT.

On Use case 3.2, the Sender is located on TestBench

5.1.1.2 Specific Requirements

None

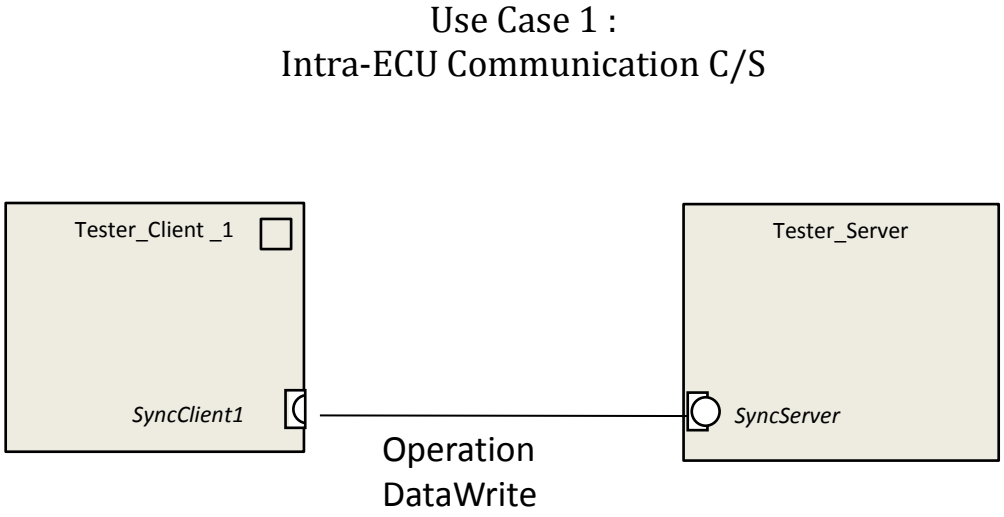
5.1.2 Test Configuration

This section describes sets of requirements on configuration.
These sets are later referenced by test cases.
No configuration files are provided. They need to be developed when the test suite is implemented.

5.1.2.1 Required ECU Extract of System Description Files

The required ECU Extract files depend on the use case:

5.1.2.1.1 Use case 03.01: intra-ECU C/S data conversion



Tests cases will use the following types, computation methods and port interface mapping:

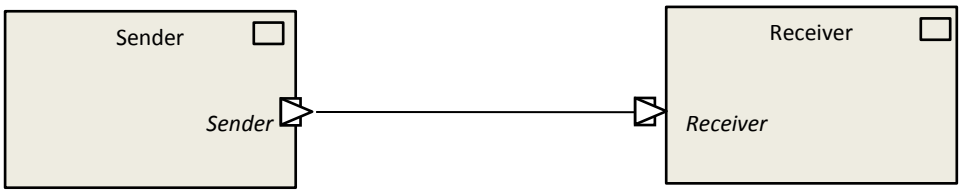
test case	ClientType/CompuMethod application data type	ServerType/CompuMethod	PortArgumentMapping
ATS_RTE_00150	km.h-1	m.s-1	
ATS_RTE_00160	0..300 km.h-1 350 -> SensorError 351 -> SignalNotAvailable	0..300 km.h-1 400 -> UnknownSpeedValue	0..300 -> 0..300 SensorError -> UnknownSpeedValue SignalNotAvailable -> UnknownSpeedValue
ATS_RTE_00145	LowerLimit : 0 UpperLimit : 100 Unit m.s-1 IntoPhy : identical	LowerLimit :200 UpperLimit : 1200 Unit m.s-1 PhyToInt = 10*Phy + 200	

ATS_RTE_00154	WheelSpeed4
	{
	Left_front : uint16;
	Left_rear:uint16;
	Right_front : uint16;
	Right_Rear: uint16;
	}
		WheelSpeed2
		{
		Left: uint16;
		Right : uint16;
		}
			WheelSpeed4.left_front -> WheelSpeed2.left
			WheelSpeed4.right_front -> WheelSpeed2.right

ATS_RTE_00158	SenderArray [4] of uint8	ReceiverArray [8] of uint8	SenderArray[0] -> ReceiverArray[4] SenderArray[1] -> ReceiverArray[5] SenderArray[2] -> ReceiverArray[6] SenderArray[3] -> ReceiverArray[7]
ATS_RTE_00149	Enum: CLIENT0, CLIENT1, CLIENT2, CLIENT3	Enum: SERVER0, SERVER1, SERVER2	TextTableMapping: CLIENT0 -> SERVER0 CLIENT1 -> SERVER2 CLIENT2 -> SERVER1 CLIENT3 -> SERVER2

5.1.2.1.2 Use case 03.02: intra-ECU S/R data conversion

Use Case 2 :
Intra-ECU Communication S/R



Tests cases will use the following types, computation methods and port interface mapping:

test case	SenderType/CompuMethod application data type	ReceiverType/CompuMethod	DataElementMapping
ATS_RTE_00146	km.h-1	m.s-1	

ATS_RTE_00160	0..300 km.h-1 350 -> SensorError 351 -> SignalNotAvailable	0..300 km.h-1 400 -> UnknownSpeedValue	0..300 -> 0..300 SensorError -> UnknownSpeedValue SignalNotAvailable -> UnknownSpeedValue
ATS_RTE_00161	LowerLimit : 0 UpperLimit : 100 Unit m.s-1 IntoPhy : identical	LowerLimit :200 UpperLimit : 1200 Unit m.s-1 PhyToInt = 10*Phy + 200	

ATS_RTE_00162	WheelSpeed4
	{
	Left_front : uint16;
	Left_rear:uint16;
	Right_front : uint16;
	Right_Rear: uint16;
	}
		WheelSpeed2
		{
		Left: uint16;
		Right : uint16;
		}
			WheelSpeed4.left_front -> WheelSpeed2.left
			WheelSpeed4.right_front -> WheelSpeed2.right

ATS_RTE_00163	SenderArray [4] of uint8	ReceiverArray [8] of uint8	SenderArray[0] -> ReceiverArray[4] SenderArray[1] -> ReceiverArray[5] SenderArray[2] -> ReceiverArray[6] SenderArray[3] -> ReceiverArray[7]
ATS_RTE_00164	Enum: SenderValue0, SenderValue1, SenderValue2, SenderValue3	Enum: ReceiverValue0, ReceiverValue1, ReceiverValue2	TextTableMapping: SenderValue0 -> ReceiverValue0 SenderValue1 -> ReceiverValue2 SenderValue2 -> ReceiverValue1 SenderValue3 -> ReceiverValue2

Each test case will ensure that the conversion have been done properly inside the generated RTE.

There are no specific requirements for these tests cases regarding the runnables and events.

5.1.2.1.3 Use case 03.03: Inter-ECU Network Representations

Tests cases will use the following types, computation methods and port interface mapping:

test case	Sender Type /compuMethod	Network Representation
ATS_RTE_00165/ATS_RTE_00166	T_VoltageAtSender uint16 FixPoint Representation with Offset : 0 LSB = 2^{-2}	T_VoltageOnNetwork uint8 FixPoint Representation with Offset = 0.5 and LSB = 2^{-1}

test case	Receiver Type /compuMethod	Network Representation
ATS_RTE_00167/ATS_RTE_00168	T_VoltageAtReceiver uint16 FixPoint Representation with Offset : 2 LSB = 2^{-3}	T_VoltageOnNetwork uint8 FixPoint Representation with Offset = 0.5 and LSB = 2^{-1}

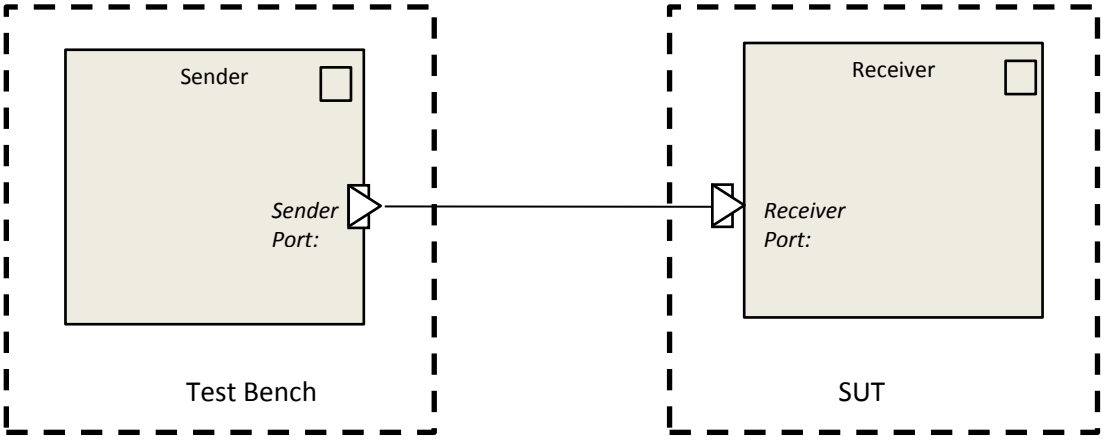
5.1.2.1.4 Use case 03.04: Intra-ECU Range Checks

The Use case 4 intends to cover the range checks intra-ECU.

This use case will reuse the SWC requirements described for Use Case 2.

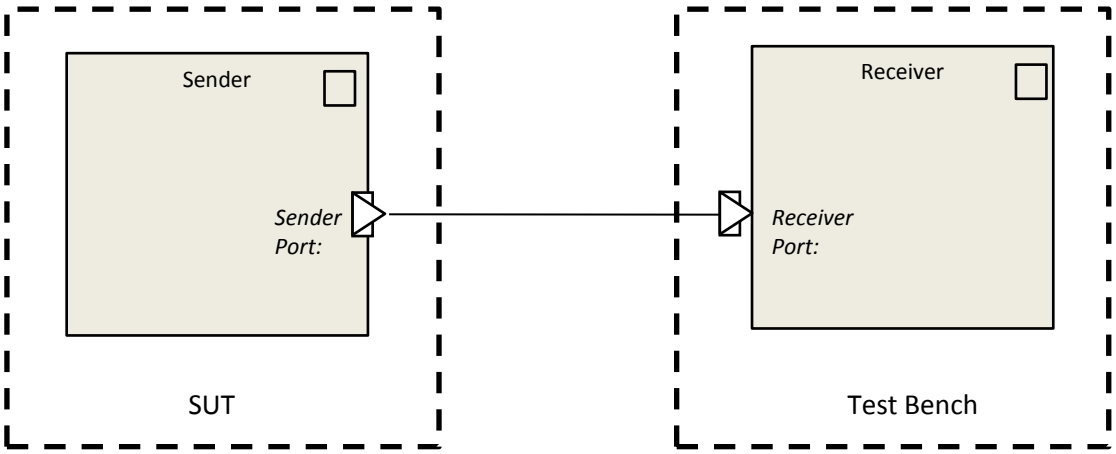
5.1.2.1.5 Use case 03.05: Inter-ECU Range Checks

Use case 5.1: inter-ECU Range Checks
Receiver on SUT



test case	Sender Type	Signals
ATS_RTE_ 00170	uint16	1 ISignal uint16 INVALID_VALUE = 0xFFFF HandleOutOfRange = Invalid

Use case 5.2: inter-ECU Range Checks
Sender on SUT



test case	Receiver Type	Signals
ATS_RTE_00176	uint16	1 ISignal uint16 INVALID_VALUE = 0xFFFF HandleOutOfRange = Invalid

5.1.2.2 Required ECU Configuration Description Files

No specific configuration requirements for ECU Configuration files as they can be derived from EcuExtract.

5.1.2.3 Required Software Component Description Files

Requirements on on Software Component Description files are provided in the section 5.1.2.1 Required ECU Extract of System Description Files, together with the connections of the different components.

5.1.2.4 Mandatory vs. Customizable Parts

Not Applicable.

5.1.3 Test Case Design

Not Applicable.

5.2 Re-usable Test Steps

Not Applicable.

5.3 Test Cases

5.3.1 [ATS_RTE_00145] Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Linear Scaling

Test Objective	Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Linear Scaling		
ID	ATS_RTE_00145	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03818 RTE: SWS_Rte_03819 RTE: SWS_Rte_03829		
Requirements / Reference to Test Environment	Use Case 03.01 : Intra-ECU C/S Communication		
Configuration Parameters	1 SWC Client The Operation uses parameter with ClientType LowerLimit = 0, UpperLimit = 100 ComputationMethod : PhytoInt : identical 1 SWC Server The Operation uses parameter with ServerType LowerLimit = 200, UpperLimit = 1200 Computation Method: PhyToInt : Linear (10*x+200) Both are using uint32 types 1 ClientServerInterfaceMapping maps the client to the server		
Summary	The Test Manager starts the Client, which calls the server The Test Manager checks that server was invoked with the converted values.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start Tester_Client_1		
Step 2	**[SWC<Tester_Client_1>]** invoke the operation (Rte_call) with argument value 0		**[SWC<Tester_Server>]** server has been invoked with converted argument value 200
Step 3	**[SWC<Tester_Client_1>]** invoke the operation (Rte_call) with argument value 100		**[SWC<Tester_Server>]** server has been invoked with converted argument value 1200
Step 5	**[CP]** terminate Tester_Client_1		

Post-conditions	None

5.3.2 [ATS_RTE_00146] Test Intra-ECU S/R dataelements rescaling - Linear Scaling different units

Test Objective	Test Intra-ECU S/R dataelements rescaling - Linear Scaling different units		
ID	ATS_RTE_00146	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03829 RTE: SWS_Rte_03832		
Requirements / Reference to Test Environment	Use Case 03.02 : Intra-ECU Communication Units conversion		
Configuration Parameters	1 SWC Sender 1 Port interface with DataElement uint32 type unit km.h-1 1 SWC Receiver 1 Port interface with DataElement uint32 type unit m.s-1		
Summary	For different values, the Test Manager starts the sender, which sends a data in km.h^-1. Then the Test Manager starts the receiver and checks that it received with the converted value.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		
Step 2	**[SWC<SWC_Sender>]** send data value 0 (km.h-1)		
Step 3	**[CP]** start SWC_Receiver		**[SWC<SWC_Receiver>]** data value 0 (m.s-1) has been received
Step 4	**[SWC<SWC_Sender>]** send data value 1 (km.h-1)		
Step 5	**[CP]** start SWC_Receiver		**[SWC<SWC_Receiver>]**

		data value 0 (m.s-1) has been received
Step 6	**[SWC<SWC_Sender>]** send data value 10 (km.h-1)	
Step 7	**[CP]** start SWC_Receiver	**[SWC<SWC_Receiver>]** data value 3 (m.s-1) has been received
Step 8	**[SWC<SWC_Sender>]** send data value 100 (km.h-1)	
Step 9	**[CP]** start SWC_Receiver	**[SWC<SWC_Receiver>]** data value 28 (m.s-1) has been received
Step 10	**[SWC<SWC_Sender>]** send data value 200 (km.h-1)	
Step 11	**[CP]** start SWC_Receiver	**[SWC<SWC_Receiver>]** data value 56 (m.s-1) has been received
Step 12	**[SWC<SWC_Sender>]** send data value 300 (km.h-1)	
Step 13	**[CP]** start SWC_Receiver	**[SWC<SWC_Receiver>]** data value 83 (m.s-1) has been received
Post-conditions	None	

5.3.3 [ATS_RTE_00149] Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Texttable to Texttable

Test Objective	Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Texttable to Texttable		
ID	ATS_RTE_00149	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03818 RTE: SWS_Rte_03830		
Requirements / Reference to Test Environment	Use case 03.01: C/S Communication		

Configuration Parameters	1 SWC Client with ENUM parameter: enum {CLIENT0,CLIENT1,CLIENT2, CLIENT3} 1 SWC Server with ENUM parameter enum {SERVER2, SERVER1, SERVER0} TextTableMapping : CLIENT0 -> SERVER0 CLIENT1 -> SERVER2 CLIENT2 -> SERVER1 CLIENT3 -> SERVER2	
Summary	The Test Manager starts the Client, which invokes a server operation with CLIENT0, then CLIENT1, then CLIENT2, then CLIENT3 values. The Test Manager checks that server has received the converted argument value on the following sequence: SERVER0, SERVER2, SERVER1, SERVER2	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Client	
Step 2	**[SWC<SWC_Client>]** invoke operation (Rte_call) with argument value CLIENT0	**[SWC<SWC_Server>]** server has been invoked with argument value SERVER0
Step 3	**[SWC<SWC_Client>]** invoke operation (Rte_call) with argument value CLIENT1	**[SWC<SWC_Server>]** server has been invoked with argument value SERVER2
Step 4	**[SWC<SWC_Client>]** invoke operation (Rte_call) with argument value CLIENT2	**[SWC<SWC_Server>]** server has been invoked with argument value SERVER1
Step 5	**[SWC<SWC_Client>]** invoke operation (Rte_call) with argument value CLIENT3	**[SWC<SWC_Server>]** server has been invoked with argument value SERVER2
Step 6	**[CP]** terminate SWC_Client	
Post-conditions	None	

5.3.4 [ATS_RTE_00150] Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Linear Scaling different units

Test Objective	Test Intra-ECU C/S argument rescaling - ClientServerInterfaceMapping Linear Scaling different units		
ID	ATS_RTE_00150	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03818 RTE: SWS_Rte_03829		
Requirements / Reference to Test Environment	Use case 03.01 : C/S		
Configuration Parameters	1 Client using parameter in m.s-1 unit (SI) 1 Server using km.h-1 unit (with associated CompuMethod defined in arxml for m.s-1 to km.h-1 conversion)		
Summary	The Client calls the operation using parameter in m.s-1 unit (SI) The test shall validate that the conversion is perfectly handled in the RTE generated code: the server is invoked using parameter in km.h-1 unit		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[CP]** start SWC_Client		
Step 2	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 0 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 0 (km.h-1)	
Step 3	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 1 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 4 (km.h-1)	
Step 4	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 10 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 36 (km.h-1)	
Step 5	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 30 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 108 (km.h-1)	

Step 6	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 60 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 216 (km.h-1)
Step 8	**[SWC<SWC_Client>]** invoke operation (Rte_Call) with argument value 85 (m.s-1)	**[SWC<SWC_Server>]** server has been invoked with argument value 306 (km.h-1)
Step 9	**[CP]** terminate SWC_Client	
Post-conditions	None	

5.3.5 [ATS_RTE_00151] Test Intra-ECU C/Sargument rescaling - C/S ApplicationErrorMapping

Test Objective	Test Intra-ECU C/Sargument rescaling - C/S ApplicationErrorMapping		
ID	ATS_RTE_00151	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_07925 RTE: SWS_Rte_07926		
Requirements / Reference to Test Environment	Use case 03.01 : C/S		
Configuration Parameters	1 SWC Client Interface : Std_ReturnType DataWriteError (uint8 DATA) The error codes for the client are : ERRORCODE2 : 2 ERRORCODE63: 63 On the server side only ERRORCODE2 is handled. ClientServerApplicationErrorMapping: Mapping ERRORCODE63 -> ERRORCODE2		
Summary	This test case validates the C/S Application Error Mapping feature provided by the RTE. For this test case, server returns ERRORCODE2 value for error occured ERRORCODE63.		
Needed Adaptation to other Releases	**Needed Adaptation for Release [4.0.3]** Configuration: [n/a] — Client-server application Error Mapping not supported. Test Steps: [n/a] — This test case shall be removed		
Pre-conditions	None		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	**[CP]** Start SWC_Client	
Step 2	**[SWC<SWC_Client>]** Call DataWriteError operation	**[SWC<SWC_Server>]** server has been invoked
Step 3	**[SWC<SWC_Server>]** Cause an error and return ERRORCODE2	**[SWC<SWC_Client>]** client has received the ERRORCODE63 return value.
Post-conditions	None	

5.3.6 [ATS_RTE_00154] Test intra-ECU C/S argument rescaling - Composite Types (Structure)

Test Objective	Test intra-ECU C/S argument rescaling - Composite Types (Structure)		
ID	ATS_RTE_00154	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03860 RTE: SWS_Rte_07038		
Requirements / Reference to Test Environment	Use Case 03.01 : Intra-ECU C/S Communication		

Configuration Parameters	1 SWC Client with argument Composite Type:
	WheelSpeed4
	{
	left_front : uint16;
	right_front : uint16;
	left_rear : uint16;
	right_rear : uint16;
	}
	1 SWC Server with following type:
	WheelSpeed2
	{
	left: unit16;
	right: uint16;
	}
	1 DataPrototypeMapping with SubElementMappings
	WheelSpeed4.left_front -> WheelSpeed2.left;
	WheelSpeed4.right_front -> WheelSpeed2.right;		

Summary	The client calls an operation with an argument as a structure with 4 fields The Test Manager shall check that server is invoked with the correct argument: a structure with 2 fields.		

Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Client	
Step 2	**[SWC<SWC_Client>]** invoke operation with composite data: WheelSpeed4.left_front = 0x55AA WheelSpeed4.right_front = 0xAA55 WheelSpeed4.left_rear = 0x5A5A WheelSpeed4.right_rear = 0xA5A5	**[SWC<SWC_Server>]** server has been invoked with composite data: WheelSpeed2.left = 0x55AA WheelSpeed2.right = 0xAA55
Step 3	**[CP]** terminate SWC_Client	
Post-conditions	None	

5.3.7 [ATS_RTE_00158] Test intra-ECU C/S argument rescaling - Composite Type (Array)

Test Objective	Test intra-ECU C/S argument rescaling - Composite Type (Array)		
ID	ATS_RTE_00158	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03860 RTE: SWS_Rte_07038		
Requirements / Reference to Test Environment	Use case 03.01 : C/S		
Configuration Parameters	1 SWC Client with Composite Array Type argument ClientArray[4] of uint8 1 SWC Server with Composite Array Type argument ServerArray[8] of uint8 1 DataPrototypeMapping with SubElementMappings ClientArray [0] -> ServerArray[4] ClientArray [1] -> ServerArray[5] ClientArray [2] -> ServerArray[6] ClientArray [3] -> ServerArray[7]		
Summary	Test Manager starts the Client which call the server with the ClientArray argument: ClientArray[0]=0x55 ClientArray[1]=0xAA		

	ClientArray[2]=0x55 ClientArray[3]=0xAA Test Manager checks that the server is invoked with the ServerArray argument ServerArray[4] = 0x55 ServerArray[5] = 0xAA ServerArray[6] = 0x55 ServerArray[7] = 0xAA
Needed Adaptation to other Releases	
Pre-conditions	None
Main Test Execution	

Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Client	
Step 2	**[SWC<SWC_Client>]** invoke operation with array argument: ClientArray[0]=0x55 ClientArray[1]=0xAA ClientArray[2]=0x55 ClientArray[3]=0xAA	**[SWC<SWC_Server>]** server has been invoked with array argument: ServerArray[4] = 0x55 ServerArray[5] = 0xAA ServerArray[6] = 0x55 ServerArray[7] = 0xAA
Step 3	**[CP]** terminate SWC_Client	
Post-conditions	None	

5.3.8 [ATS_RTE_00160] Test intra-ECU S/R dataelements rescaling - Mixed Linear scaled and texttable

Test Objective	Test intra-ECU S/R dataelements rescaling - Mixed Linear scaled and texttable		
ID	ATS_RTE_00160	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03815 RTE: SWS_Rte_03816 RTE: SWS_Rte_03832 RTE: SWS_Rte_03855		
Requirements / Reference to Test Environment	use Case 03.02: Intra-ECU S/R Communication		
Configuration Parameters	1 SWC Sender Sender Type: SpeedSensorType with SCALE_LINEAR_AND_TEXTTABLE CompuMethod:		

	values linear 0..300 value 350 -> SensorError value 351 -> SignalNotAvailable units : km.h-1 1 SWC Receiver Receiver Type: SpeedType with SCALE_LINEAR_AND_TEXTTABLE CompuMethod: values linear 0..350 value 400 -> UnknowSpeedValue units : km.h-1 DataPrototypeMapping between those AutosarDataPrototypes with TextTableMappings value 350 (SensorError)-> 400 (UnknownSpeedValue) value 351 (SignalNotAvailable) -> 400 (UnknownSpeedValue) Both types are based on uint16
Summary	The test shall validate that the conversion is perfectly handled in the RTE generated code for different values: 1. Send the value 150 and check that Receiver receives the value 150 in Speeddataelement 2. Send the value SensorError and check that Receiver receives the value UnknownSpeedValue 3. Send the value SignalNotAvailable and check that Receiver receives the value UnknownSpeedValue
Needed Adaptation to other Releases	
Pre-conditions	None
Main Test Execution	

Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Sender	
Step 2	**[SWC<SWC_Sender>]** send (Rte_Write) value 150 (km.h-1)	**[SWC<SWC_Receiver>]** receiver has received data value 150 (km.h-1)
Step 3	**[SWC<SWC_Sender>]** send (Rte_Write) value 350 (SensorError)	**[SWC<SWC_Receiver>]** receiver has received data value 400 (UnknownSpeedValue)
Step 4	**[SWC<SWC_Sender>]** send (Rte_Write) value 351 (SignalNotAvailable)	**[SWC<SWC_Receiver>]** receiver has received data value 400 (UnknownSpeedValue)
Step 5	**[CP]** terminate SWC_Sender	

Post-conditions	None

5.3.9 [ATS_RTE_00161] Test intra-ECU S/R dataelements rescaling - Linear Scaled conversion

Test Objective	Test intra-ECU S/R dataelements rescaling - Linear Scaled conversion		
ID	ATS_RTE_00161	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03815 RTE: SWS_Rte_03829		
Requirements / Reference to Test Environment	Use case 03.02: Intra-ECU Communication S/R		
Configuration Parameters	1 SWC Sender with SenderReceiverInterface dataElement typed with SenderType LowerLimit = 0, UpperLimit = 100 ComputationMethod : PhytoInt : identical 1 SWC Receiver with dataElement typed with ReceiverType LowerLimit = 200, UpperLimit = 1200 Computation Method: PhyToInt : Linear (10*x+200) 1 DataPrototypeMapping between those AutosarDataPrototypes Both types are based on uint32 Computation Method: PhyToInt : Linear (10*x+200)		
Summary	For different values, a value is sent from SWC_Sender and SWC_Receiver checks it received the converted value.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		
Step 2	**[SWC<SWC_Sender>]** send (Rte_Write) DataWrite with data value 0		**[SWC<SWC_Receiver>]** receiver has received converted data value 200
Step 3	**[SWC<SWC_Sender>]** send (Rte_Write) DataWrite with data value 100		**[SWC<SWC_Receiver>]** receiver has received converted data value 1200
Step 5	**[CP]** terminate SWC_Sender		

Post-conditions	None

5.3.10 [ATS_RTE_00162] Test intra-ECU S/R dataelements rescaling - Composite Types (Structure)

Test Objective	Test intra-ECU S/R dataelements rescaling - Composite Types (Structure)		
ID	ATS_RTE_00162	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03815 RTE: SWS_Rte_03816 RTE: SWS_Rte_03860 RTE: SWS_Rte_07091 RTE: SWS_Rte_07092 RTE: SWS_Rte_07099		
Requirements / Reference to Test Environment	Use case 03.02 : S/R		

Configuration Parameters	1 SWC Sender with a Sender interface :
	WheelSpeed4
	{
	left_front : uint16;
	left_rear : unit16;
	right_front : uint16;
	right_rear : uint16;
	}
	1 SWC receiver with the following interface:
	WheelSpeed2
	{
	left : uint16;
	right : uint16;
	}
	Mapping:
	WheelSpeed4.left_front -> WheelSpeed2.left
	WheelSpeed4.right_front -> WheelSpeed2.right		

Summary	This test case validates the feature provided by the RTE of sender-receiver dataelements mapping of Composite types like C structures. The Test Manager shall check that CompositeData has been converted correctly by the RTE on the receiver side.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	**[CP]** start SWC_Sender	
Step 2	**[SWC<SWC_Sender>]** send (Rte_Write) composite data: WheelSpeed4.left_front = 0x55AA WheelSpeed4.right_front = 0xAA55 WheelSpeed4.left_rear = 0x5A5A WheelSpeed4.right_rear = 0xA5A5	**[SWC<SWC_Receiver>]** receiver has received composite data: WheelSpeed2.left = 0x55AA WheelSpeed2.right = 0xAA55
Step 3	**[CP]** terminate SWC_Sender	
Post-conditions	None	

5.3.11 [ATS_RTE_00163] Test intra-ECU S/R dataelements rescaling - Composite Types (Array)

Test Objective	Test intra-ECU S/R dataelements rescaling - Composite Types (Array)		
ID	ATS_RTE_00163	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03815 RTE: SWS_Rte_03816 RTE: SWS_Rte_03860 RTE: SWS_Rte_07091 RTE: SWS_Rte_07092 RTE: SWS_Rte_07099		
Requirements / Reference to Test Environment	Use case 03.02: S/R		
Configuration Parameters	A SWC sender Array uint8 SenderArray[4] A Receiver Array uint8 ReceiverArray[8] Using SubElementMappings, the array elements are mapped as follows: SenderArray[0] -> ReceiverArray[4] SenderArray[1] -> ReceiverArray[5] SenderArray[2] -> ReceiverArray[6] SenderArray[3] -> ReceiverArray[7]		
Summary	The sender runnable sends the following array SenderArray[0]=0x55 SenderArray[1]=0xAA SenderArray[2]=0x55 SenderArray[3]=0xAA		

	Test Manager checks that the receiver runnable receives an array with ReceiverArray[4] = 0x55 ReceiverArray[5] = 0xAA ReceiverArray[6] = 0x55 ReceiverArray[7] = 0xAA
Needed Adaptation to other Releases	
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Sender	
Step 2	**[SWC<SWC_Sender>]** send array: SenderArray[0]=0x55 SenderArray[1]=0xAA SenderArray[2]=0x55 SenderArray[3]=0xAA	**[SWC<SWC_Receiver>]** the following array has been received: ReceiverArray[4] = 0x55 ReceiverArray[5] = 0xAA ReceiverArray[6] = 0x55 ReceiverArray[7] = 0xAA
Step 3	**[CP]** terminate SWC_Sender	
Post-conditions	None	

5.3.12 [ATS_RTE_00164] Test intra-ECU S/R dataelements rescaling - texttable to texttable

Test Objective	Test intra-ECU S/R dataelements rescaling - texttable to texttable		
ID	ATS_RTE_00164	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00028		
Trace to SWS Item	RTE: SWS_Rte_03815 RTE: SWS_Rte_03816 RTE: SWS_Rte_03830 RTE: SWS_Rte_03833		
Requirements / Reference to Test Environment	Use case 03.02 : S/R		
Configuration Parameters	1 SWC Sender with ENUM parameter: enum		

	{SenderValue0,SenderValue1,SenderValue2,SenderValue3} 1 SWC Receiver with ENUM parameter enum {ReceiverValue2, ReceiverValue1, ReceiverValue0} TextTableMapping : SenderValue0 -> ReceiverValue0 SenderValue1 -> ReceiverValue2 SenderValue2 -> ReceiverValue1 SenderValue3 -> ReceiverValue2
Summary	Test Manager starts the SWC Sender Sender sends to Receiver SenderValue0, then SenderValue1, then SenderValue2, then SenderValue3 values. Receiver checks that it has received the converted value on the following sequence: ReceiverValue0, ReceiverValue2, ReceiverValue1, ReceiverValue2
Needed Adaptation to other Releases	
Pre-conditions	None
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	**[CP]** start SWC_Sender
Step 2	**[SWC<SWC_Sender>]** send value SenderValue0	**[SWC<SWC_Receiver>]** ReceiverValue0 has been received
Step 3	**[SWC<SWC_Sender>]** send value SenderValue1	**[SWC<SWC_Receiver>]** ReceiverValue2 has been received
Step 4	**[SWC<SWC_Sender>]** send value SenderValue2	**[SWC<SWC_Receiver>]** ReceiverValue1 has been received
Step 5	**[SWC<SWC_Sender>]** send value SenderValue3	**[SWC<SWC_Receiver>]** ReceiverValue2 has been received
Step 6	**[CP]** terminate SWC_Sender	
Post-conditions	None	

5.3.13 [ATS_RTE_00165] Test Inter-ECU NetworkRepresentations - Transmitting ECU DatatTypePolicy : Override

Test Objective	Test Inter-ECU NetworkRepresentations - Transmitting ECU DatatTypePolicy : Override		
ID	ATS_RTE_00165	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2

Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00027		
Trace to SWS Item	RTE: SWS_Rte_03827		
Requirements / Reference to Test Environment	Use Case 03.03: Inter-ECU Network Representations		
Configuration Parameters	SWC_Sender sends a dataelement of a sint16 type t_VoltageAtSender SenderPort representation called s Physical Representation called p CompuMethod InternalToPhysical : p=(0+1*s)/4 PhysicalToInternal : s=4p NetworkRepresentation (called n) CompuMethod InternalToPhysical : p=(1+n)/2 PhysicalToInternal : n=2p-1		
Summary	For this test case, the SWC_Sender is located on the SUT Side. The SWC_Receiver is located on the Test bench side.. The Test Bench shall check that transmitted data over the network has been received correctly and has been converted as defined.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		
Step 2	**[SWC<SWC_Sender>]** send a data value 0		**[LT]** data value -1 (=(0/2)-1) has been received
Step 3	**[SWC<SWC_Sender>]** send a data value 10		**[LT]** data value 4 (=(10/2)-1) has been received
Step 4	**[SWC<SWC_Sender>]** send a data value 100		**[LT]** data value 49 (=(100/2)-1) has been received
Step 5	**[CP]** terminates SWC_Sender		

Post-conditions	None

5.3.14 [ATS_RTE_00166] Test Inter-ECU NetworkRepresentations - Transmitting ECU DatatTypePolicy : FromComSpec

Test Objective	Test Inter-ECU NetworkRepresentations - Transmitting ECU DatatTypePolicy : FromComSpec		
ID	ATS_RTE_00166	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00027		
Trace to SWS Item	RTE: SWS_Rte_06737		
Requirements / Reference to Test Environment	Use Case 03.03: inter-ECU Network Representations		
Configuration Parameters	SWC_Sender sends a dataelement of a sint16 type t_VoltageAtSender SenderPort representation called s Physical Representation called p CompuMethod InternalToPhysical : p=(0+1*s)/4 PhysicalToInternal : s=4p NetworkRepresentation (called n) CompuMethod InternalToPhysical : p=(1+n)/2 PhysicalToInternal : n=2p-1 EcuC Justification: This networkRepresentation is defined in the SenderComSpec of SWC_Sender and on the SenderComSpec of SwcB and the corresponding ISignal is defined		
Summary	This test case validates the data conversion over the network from Transmitting ECU side. This test case consist in sending the data value on the port and check that the data has been received on the Test Bench side with the correct value regarding the network representation defined.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		

Step 2	**[SWC<SWC_Sender>]** send a data value 0	**[LT]** data value -1 has been received
Step 3	**[SWC<SWC_Sender>]** send a data value 10	**[LT]** data value 4 (=(10/2)-1) has been received
Step 4	**[SWC<SWC_Sender>]** send a data value 100	**[LT]** data value 49 (=(100/2)-1) has been received
Step 5	**[CP]** terminate SWC_Sender	
Post-conditions	None	

5.3.15 [ATS_RTE_00167] Test Inter-ECU NetworkRepresentations - Receiving ECU DatatTypePolicy : Override

Test Objective	Test Inter-ECU NetworkRepresentations - Receiving ECU DatatTypePolicy : Override		
ID	ATS_RTE_00167	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00027		
Trace to SWS Item	RTE: SWS_Rte_03828		
Requirements / Reference to Test Environment	Use case 03.03: Inter-ECU Network representations		
Configuration Parameters	SwcB receives a dataelement of a sint16 type t_VoltageAtReceiver ReceiverPort representation called r Physical Representation called p CompuMethod InternalToPhysical : p=(16+r)/8 PhysicalToInternal : r=8p-16 NetworkRepresentation (called n) CompuMethod InternalToPhysical : p=(1+n)/2 PhysicalToInternal : n=2p-1 EcuC Justification: DataTypePolicy = Override		
Summary	This test case validates the Data conversion of Network transmitted data on the receiver side when ISignal DataTypePolicy is set to Override.		

	For this test case, the SwcA is located on the Test Bench Side. The SwcB is located on the SUT side. The Test Bench shall check that transmitted data over the network have been received and converted correctly by the RTE.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Receiver	
Step 2	**[LT]** sends a data value -1 on network	**[SWC<SWC_Receiver>]** data value -16 (=-1*4-12) has been received
Step 3	**[LT]** send a data value 4 on network	**[SWC<SWC_Receiver>]** data value 4 (=4*4-12) has been received
Step 4	**[LT]** send a data value 49 on network	**[SWC<SWC_Receiver>]** data value 184 (=49*4-12) has been received
Step 5	**[CP]** terminate SWC_Receiver	
Post-conditions	None	

5.3.16 [ATS_RTE_00168] Test Inter-ECU NetworkRepresentations - Receiving ECU DatatTypePolicy : FromComSpec

Test Objective	Test Inter-ECU NetworkRepresentations - Receiving ECU DatatTypePolicy : FromComSpec		
ID	ATS_RTE_00168	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00027		
Trace to SWS Item	RTE: SWS_Rte_06738		
Requirements / Reference	Use case 03.03: Inter-ECU Network Representations		

to Test Environment		
Configuration Parameters	SwcB receives a dataelement of a sint16 type t_VoltageAtReceiver ReceiverPort representation called r Physical Representation called p CompuMethod InternalToPhysical : p=(16+r)/8 PhysicalToInternal : r=8p-16 NetworkRepresentation (called n) CompuMethod InternalToPhysical : p=(1+n)/2 PhysicalToInternal : n=2p-1 EcuC Justification: DataTypePolicy = FromComSpec	
Summary	This test case validates the data conversion over the network from Receiving ECU side. This test case consist in sending the data value by the test bench and check that the data has been received on the SUT side with the correct value regarding the network representation defined.	
Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Receiver	
Step 2	**[LT]** send a data value -1 on network	**[SWC<SWC_Receiver>]** data value -16 (=4*0-12) has been received
Step 3	[LT] send a data value 4 on network	**[SWC<SWC_Receiver>]** data value 4 (=4*4-12) has been received
Step 4	**[LT]** send a data value 49 on network	**[SWC<SWC_Receiver>]** data value 184 (=49*4-12) has been received
Step 5	**[CP]** terminate SWC_Receiver	
Post-conditions	None	

5.3.17 [ATS_RTE_00169] Test Intra-ECU Range Checks - SR Unqueued ignore

Test Objective	Test Intra-ECU Range Checks - SR Unqueued ignore

ID	ATS_RTE_00169	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00029		
Trace to SWS Item	RTE: SWS_Rte_03839 RTE: SWS_Rte_03845 RTE: SWS_Rte_08026 RTE: SWS_Rte_08028		
Requirements / Reference to Test Environment	Use case 03.04: Range checks intra-ECU		
Configuration Parameters	SUT has 2 SWC Data Element Type defined with: uint16 LowerLimit : 0 UpperLimit : 100 SwcA with SenderPort with UnqueuedSenderComSpec defined with HandleOutOfRange = Ignore SwcB with ReceiverPort with UnqueuedReceiverComSpec defined with handleOutOfRange = ignore		
Summary	This test will send different values and check that received data on the receiver side has been saturated to the defined UpperLimit (100).		
Needed Adaptation to other Releases			
Pre-conditions	None		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** start SWC_Sender	
Step 2	**[SWC<SWC_Sender>]** send data value 100	**[SWC<SWC_Receiver>]** data value 100 has been received
Step 3	**[SWC<SWC_Sender>]** send data value 101	**[SWC<SWC_Receiver>]** no data has been received hint: scheduling must be controlled so that enough time is given to receive a data (as in step 2)
Step 4	**[SWC<SWC_Sender>]** send data value 200	**[SWC<SWC_Receiver>]** no data has been received hint: scheduling must be controlled so that enough time is given to receive a data (as in step 2)
Step 5	**[CP]** terminate SWC_Sender	

Post-conditions	None

5.3.18 [ATS_RTE_00170] Test Inter-ECU Range Checks - Sender Side ISignalProps = invalid

Test Objective	Test Inter-ECU Range Checks - Sender Side ISignalProps = invalid		
ID	ATS_RTE_00170	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00029		
Trace to SWS Item	RTE: SWS_Rte_08033		
Requirements / Reference to Test Environment	Use Case 03.05: Range Checks Inter-ECU		
Configuration Parameters	1 SWC Sender on the SUT 1 SenderPort with data element uint16 LowerLimit : 0 Upperlimit : 100 (these limits should be defined on the networkRepresentationProps of the ISignal (SWS_Rte_08042), not on the dataElement) 1 ISignal uint16 networkRepresentationProps with invalidValue INVALID_VALUE : 0xFFFF ISignalProps with handleOutOfRange = hnvalid		
Summary	The Test case will send different data and check that for data outof range on the SWC Sender, the INVALID_VALUE is received by SWC Receiver.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		
Step 2	**[SWC<SWC_Sender>]** send data value 100		**[LT]** data value 100 has been received
Step 3	**[SWC<SWC_Sender>]** send data value 101		**[LT]** data value INVALID_VALUE has been received
Step 4	**[CP]** terminate SWC_Sender		
Post-conditions	None		

5.3.19 [ATS_RTE_00175] Test Intra-ECU Range Checks - SR queued saturate

Test Objective	Test Intra-ECU Range Checks - SR queued saturate		
ID	ATS_RTE_00175	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00029		
Trace to SWS Item	RTE: SWS_Rte_03840 RTE: SWS_Rte_08026		
Requirements / Reference to Test Environment	Use case 03.04: Intra-ECU Range checks		
Configuration Parameters	SUT has 2 SWC Data Element Type defined with: uint16 isQueued=true LowerLimit : 0 UpperLimit : 100 SwcA with SenderPort with QueuedSenderComSpec defined with HandleOutOfRange = saturate SwcB with ReceiverPort with QueuedReceiverComSpec defined with handleOutOfRange = saturate		
Summary	This test will send different values and check that received data on the receiver side has been saturated to the defined UpperLimit (100).		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Sender		
Step 2	**[SWC<SWC_Sender>]** send (Rte_Send) data value 99 send (Rte_Send) data value 100 send (Rte_Send) data value 101		
Step 3	**[CP]** Wait SWC_Receiver activation		**[SWC<SWC_Receiver>]** data value 99 has been received (Rte_Receive)
Step 4	**[CP]** Wait SWC_Receiver activation		**[SWC<SWC_Receiver>]** data value 100 has been received (Rte_Receive)
Step 5	**[CP]** Wait SWC_Receiver activation		**[SWC<SWC_Receiver>]** data value 100 has been received

Step 6	**[CP]** terminate SWC_Sender	
Post-conditions	None	

5.3.20 [ATS_RTE_00176] Test Inter-ECU Range Checks - Receiver Side ISignalProps = invalid

Test Objective	Test Inter-ECU Range Checks - Receiver Side ISignalProps = invalid		
ID	ATS_RTE_00176	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE, COM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00029		
Trace to SWS Item	RTE: SWS_Rte_08037		
Requirements / Reference to Test Environment	Use case 03.05: Inter-ECU Range Checks		
Configuration Parameters	1 SWC Receiver on the SUT 1 ReceiverPort with data element uint16 LowerLimit : 0 Upperlimit : 100 1 (these limits should be defined on the networkRepresentationProps of the ISignal (SWS_Rte_08042), not on the dataElement) ISignal uint16 networkRepresentationProps with invalidValue INVALID_VALUE : 0xFFFF and iSignalProps with handleOutOfRange = invalid		
Summary	This test sends a ISignal on the CAN Bus from the Test Bench to SUT. The ISignal value is out of range (200) Check that on receiver side, the received value is INVALID_VALUE.		
Needed Adaptation to other Releases			
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** start SWC_Receiver		
Step 2	**[LT]** send signal with value 100 on network		**[SWC<SWC_Receiver>]** data value 100 has been received (Rte_Read) Rte_Read returns RTE_E_OK

Step 3	**[LT]** send signal with value 101 on network	**[SWC<SWC_Receiver>]** data value INVALID_VALUE has been received (Rte_Read) Rte_Read returns RTE_E_OK
Step 5	**[CP]** terminate SWC_Receiver	
Post-conditions	None	

6 RS_BRF_01304/RS_BRF_01352 – Rte Sender Receiver Communication

6.1 General Test Objective and Approach

The “RTE 1:n Sender Receiver” features are tested by sending and receiving data via SW-Cs’ ports or via the bus, and then checking if the data that has been transmitted or received correctly and with the appropriate timing.

6.1.1 Test System

6.1.1.1 Overview on Architecture

The basic test setup is depicted in Figure 4, corresponding to the test of the “1:n Sender Receiver” features, that can be refined depending on the communication pattern (intra-ECU or inter-ECU) and the number of receivers (1:1 or 1:2) required to execute the test case.

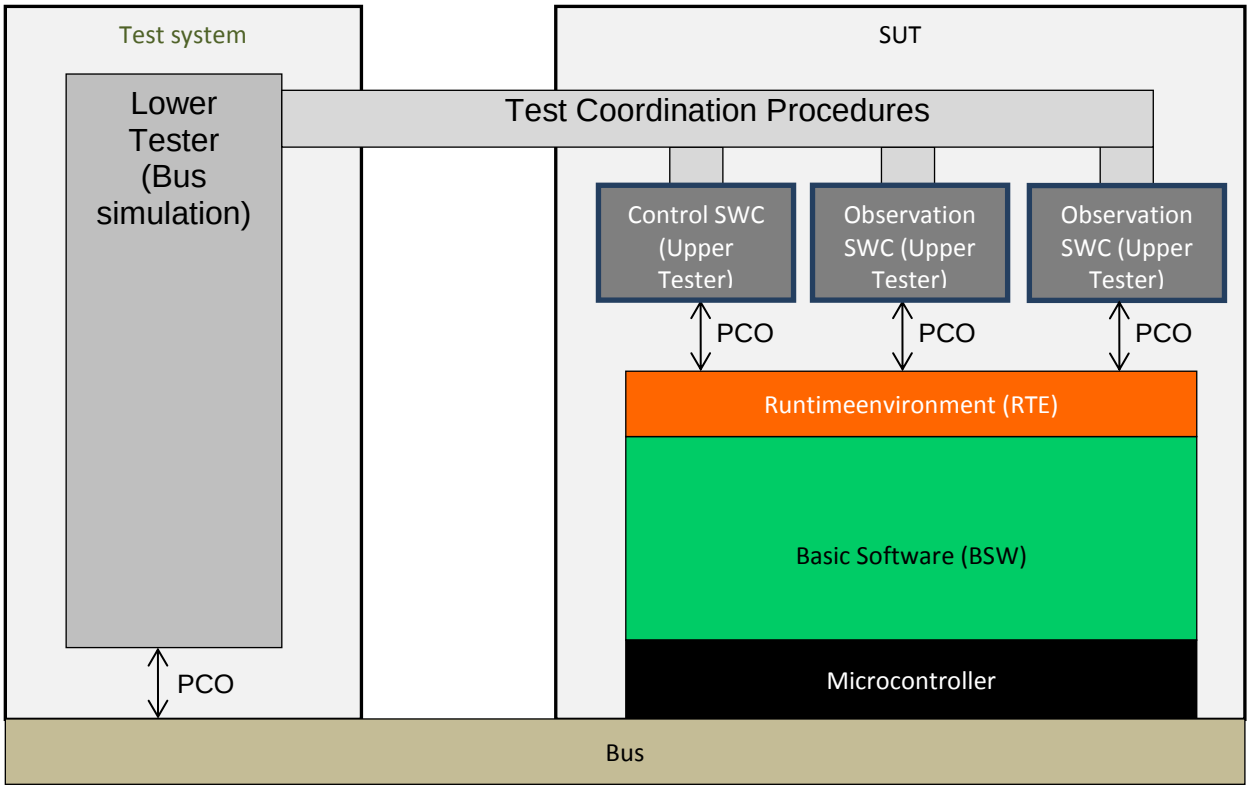


Figure 4: Basic test setup

It can be refined as follows:

6.1.1.1.1 1:1 intra-ECU test cases

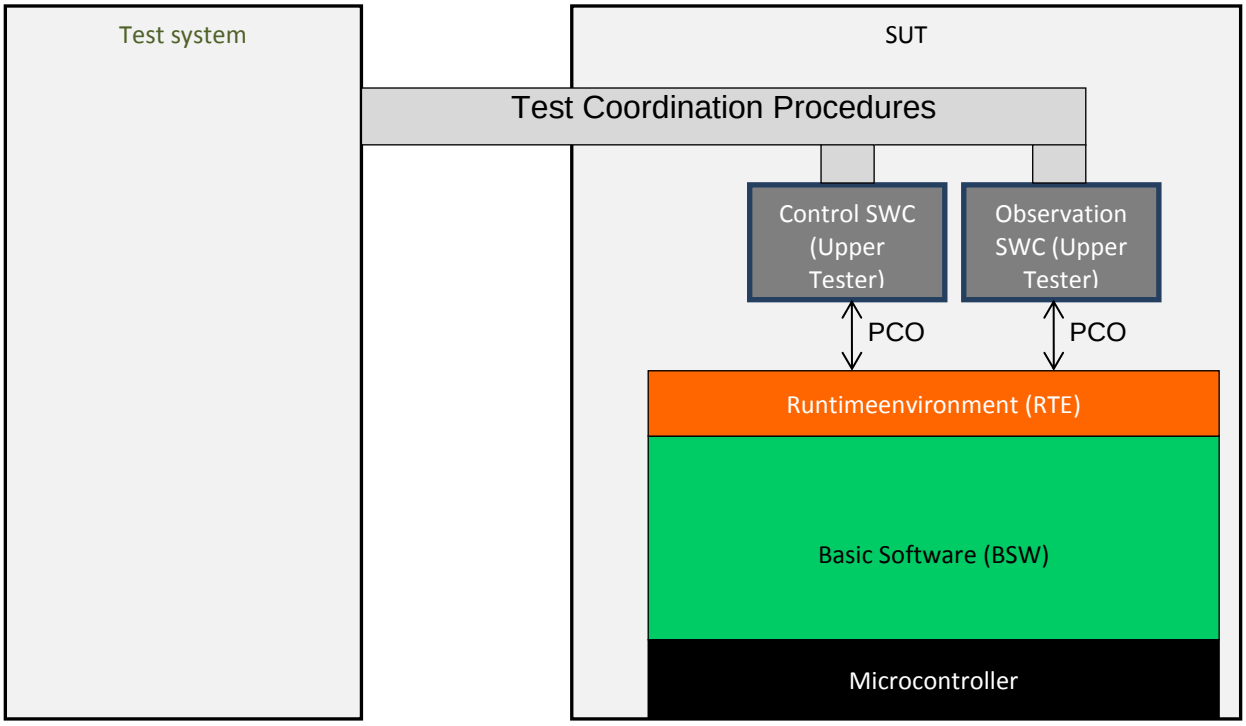


Figure 5: Test Setup for an Intra-ECU test

1:1 intra-ECU test cases require two SWCs as Upper Testers. The Control SWC is responsible for the transmission of data on a provide port, while the Observation SWC is responsible for the reception of data on a require port.

6.1.1.1.2 1:1 Inter-ECU test cases

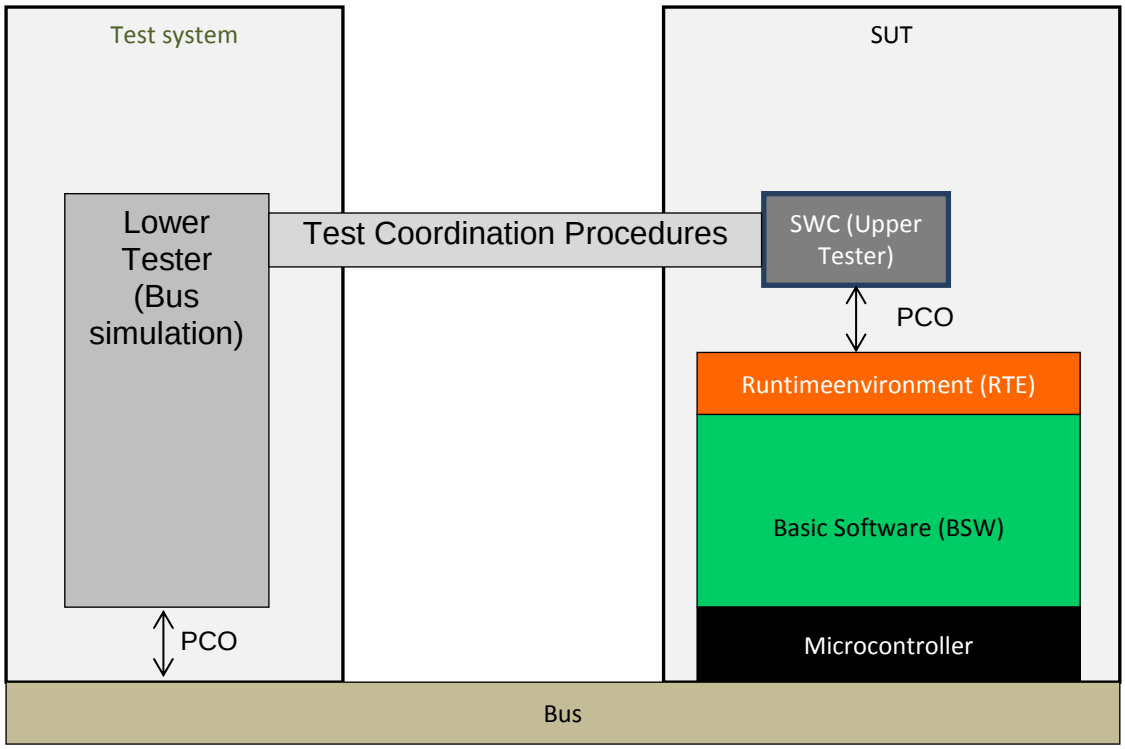


Figure 6: Test Setup for Inter-ECU Test

1:1 inter-ECU test cases require a SWC as Upper Tester. This SWC performs transmission of data on a provide port and the correct behavior of the “RTE Sender Receiver” features is verified by the Lower Tester which observes the bus transporting the data. The SWC is also responsible for the reception of data transmitted on the bus.

6.1.1.1.3 1:2 reception from inter ECU test cases

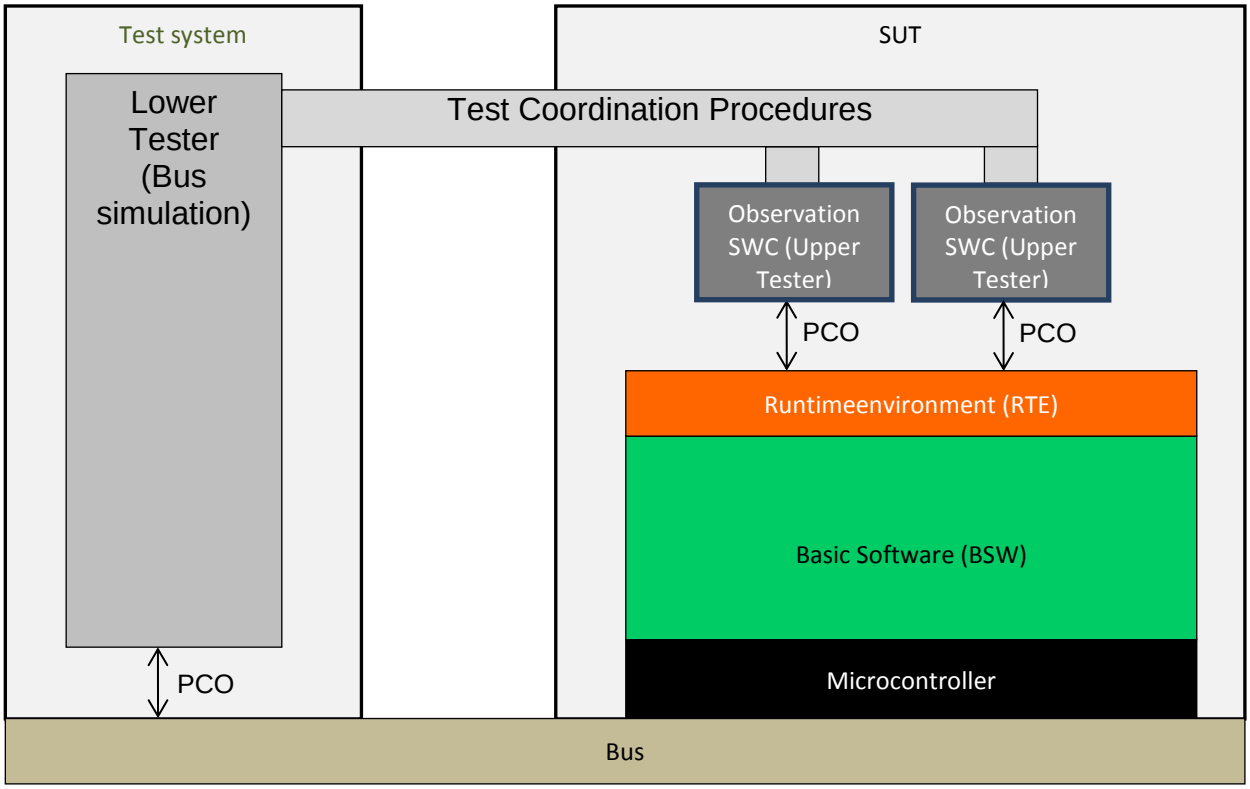


Figure 7: Test Setup for 1:2 Inter-ECU Test

1:2 reception from inter-ECU test cases require two SWCs as Upper Testers. These SWCs perform on their require ports the reception of data received by the ECU via the bus.

6.1.1.1.4 1:2 transmission to intra-ECU & inter-ECU test cases

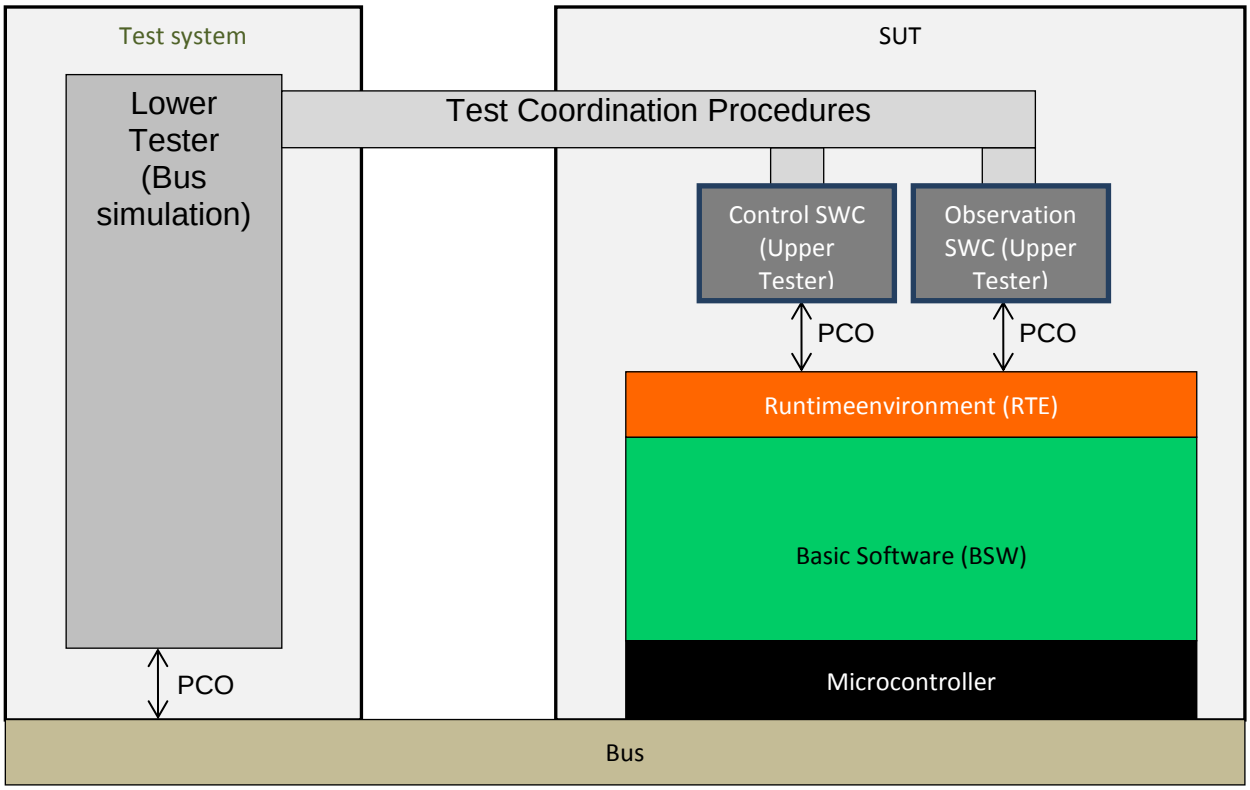


Figure 8: Test Setup for 1:2 Transmission Test

1:2 transmission to intra-ECU & inter-ECU test cases require two SWCs as Upper Testers. The Control SWC performs on a provide port transmission of data that is made available to the other SWC and on the bus. The Observation SWC performs a reception of data on the provide port while the Lower Tester receives the data via the bus.

6.1.1.2 Specific Requirements

Test cases with inter-ECU communication pattern require a “Bus simulation” as Lower Tester. It is used to simulate the reception and the transmission of messages by other ECUs.

In inter-ECU test cases, the bus must be able to transmit any messages from Lower Tester to the Observation SWC and from the Control SWC to the Lower Tester. Test cases implicitly suppose that when a message is sent on the bus, this message is effectively transmitted by the bus.

6.1.1.3 Test Coordination Requirements

Test Coordination Procedures are needed to synchronize the runnables of the Control SWC and of the Observation SWC in intra-ECU test cases.

TCPs are also needed to collect the test results of the SWCs and the Bus simulation at one central place in order to derive the test verdict.

It is up to the test system designer/implementer to define that “central place” and to design/implement the test coordination functionality.

6.1.2 Test Configuration

This section describes sets of requirements on configuration.
These sets are later referenced by test cases.
No configuration files are provided. They need to be developed when the test suite is implemented.

6.1.2.1 Required ECU Extract of System Description Files

A single ECU Extract is needed as input for the test cases described in this document.

6.1.2.1.1 EcuFlatView

This element includes:

- The ECUFlatView composition;
- The instances of the Atomic Software Components included in the SUT Ecu:
  - ControlSWC
  - Observation1SWC
  - Observation2SWC

  with their ports and their connections.

A RootSwCompositionPrototype pointing to this ECU flat view must also be included in the ECU Extract.

6.1.2.1.2 SoftwareComponent Description

The description of all the Atomic Software Components included in the SUT Ecu, with their interfaces, datatypes, internal behavior and implementation must be included.(see section 6.1.2.3)

6.1.2.1.3 Topology

This section includes:

- The instance of the SUT Ecu, with a Communication Controller to make communication possible with the Test System;
- The Communication Cluster with a physical channel used by the SUT Ecu instance.

6.1.2.1.4 Communication

To describe the communication characteristics between the SUT Ecu and the Test system, for each test case the following elements must be included:

- two System Signals for the two directions of communication (from SUT Ecu to the Test System and viceversa);
- two ISignal, each mapped to one of the defined System Signal;

- two IPdu, each mapped to one of the defined ISignals;
- two Frames, each mapped to one of the defined IPdu

During the implementation phase, it can choose to reuse some of these elements.

6.1.2.1.5 Mappings

The Ecu Extract also includes the following mappings:

- software components to SUT Ecu mapping;
- data to System Signals mapping.

Data are to be mapped to the system signals according to their direction of communication

6.1.2.2 Required ECU Configuration Specifications

There are no generic requirements on ECU Configuration files. Individual test cases may have specific requirements on the RTE configuration.

6.1.2.3 Required Software Component Specifications

Types:

Name	Type
TYP_ExchangedData	<Implementer choice>

Interfaces:

Name	Data Element	Type	SwImplPolicy
IF_ExchangedData	DE_ExchangedData	TYP_ExchangedData	standard
	DE_ExchangedEvent	TYP_ExchangedData	queued

SWCs:

SWC Name	ControlSWC	
PORTS	Name	PP_SendData
	Type	PPortPrototype
	Interface	IF_ExchangedData
	ComSpecs	NonqueuedSenderComSpec for dataElement DE_ExchangedData, with initValue Value_Init_Send.
	Requirements	
	Name	PP_SendEvent
	Type	PPortPrototype
	Interface	IF_ExchangedData
	ComSpecs	QueuedSenderComSpec for dataElement DE_ExchangedEvent.
	Requirements	
	Name	RUN_Ctrl

RUNNABLE ENTITIES	**Requirements**	Executed in the same task as RUN_Obs2_1 and RUN_Obs2_2.	
	VariableDataPrototype	**Name**	VDP_ImplWrite
		Type	DataWriteAccess
		Access to	PP_SendData/ IF_ExchangedData/ DE_ExchangedData
		Requirements	Belongs to the same coherency group as RUN_Obs2_1/ VDP_ImplRead2 and RUN_Obs2_2/ VDP_ImplRead2_2
		Name	VDP_ExplWrite
		Type	DataSendPoint
		Access to	PP_SendData/ IF_ExchangedData/ DE_ExchangedData
		Requirements	
		Name	VDP_Send
		Type	DataSendPoint
		Access to	PP_SendEvent/ IF_ExchangedData/ DE_ExchangedEvent
		Requirements	

SWC Name	Observation1SWC	
PORTS	**Name**	RP_ReceiveData1
	Type	RPortPrototype
	Interface	IF_ExchangedData
	ComSpecs	NonqueuedReceiverComSpec for dataElement DE_ExchangedData, with initValue Value_Init_Send.
	Requirements	
	Name	RP_ReceiveEvent1
	Type	RPortPrototype
	Interface	IF_ExchangedEvent
	Queuing attribute	QueuedReceiverComSpec for dataElement DE_ExchangedEvent, with queueLength = 2.
	Init value	

	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_Obs1	
	Requirements		
	VariableDataPrototype	**Name**	VDP_ImplRead1
		Type	DataReadAccess
		Access to	RP_ReceiveData1/ IF_ExchangedData/ DE_ExchangedData
		Requirements	
		Name	VDP_ExplRead1
		Type	DataReceivePoint
		Access to	RP_ReceiveData1/ IF_ExchangedData/ DE_ExchangedData
		Requirements	Non blocking
		Name	VDP_Receive1
		Type	DataReceivePoint
		Access to	RP_ReceiveEvent1/ IF_ExchangedData/ DE_ExchangedEvent
		Requirements	Non blocking

SWC Name	Observation2SWC		
PORTS	**Name**	RP_ReceiveData2	
	Type	RPortPrototype	
	Interface	IF_ExchangedData	
	ComSpecs	NonqueuedReceiverComSpec for dataElement DE_ExchangedData, with initValue Value_Init_Receive2, and with attribute handleNeverReceived set to TRUE.	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_Obs2_1	
	Requirements	Executed in the same task as RUN_Obs2_2 and RUN_Ctrl.	
	VariableDataPrototype	**Name**	VDP_ImplRead2_1
		Type	DataReadAccess
		Access to	RP_ReceiveData2/ IF_ExchangedData/ DE_ExchangedData

		Requirements	Belongs to the same coherency group as RUN_Obs2_2/ VDP_ImplRead2_2 and RUN_Ctrl/ VDP_ImplWrite
		Name	VDP_ExplRead2_1
		Type	DataReceivePoint
		Access to	RP_ReceiveData2/ IF_ExchangedData/ DE_ExchangedData
		Requirements	Non blocking
	Name	RUN_Obs2_2	
	Requirements	Executed in the same task as RUN_Obs2_1 and RUN_Ctrl.	
	VariableDataPrototype	**Name**	VDP_ImplRead2_2
		Type	DataReadAccess
		Access to	RP_ReceiveData2/ IF_ExchangedData/ DE_ExchangedData
		Requirements	Belongs to the same coherency group as RUN_Obs2_1/ VDP_ImplRead2_1 and RUN_Ctrl/ VDP_ImplWrite

SWC Name	LowerTesterSWC	
PORTS	**Name**	PP_BusSendData
	Type	PPortPrototype
	Interface	IF_ExchangedData
	ComSpecs	NonqueuedSenderComSpec for dataElement DE_ExchangedData, with initValue Value_Init_BusSend.
	Requirements	
	Name	RP_BusReceiveData
	Type	RPortPrototype
	Interface	IF_ExchangedData
	ComSpecs	NonqueuedReceiverComSpec for dataElement DE_ExchangedData, with initValue Value_Init_BusReceive.

	Requirements		
	Name	PP_BusSendEvent	
	Type	PPortPrototype	
	Interface	IF_ExchangedData	
	ComSpecs	QueuedSenderComSpec for dataElement DE_ExchangedEvent.	
	Requirements		
	Name	RP_BusReceiveEvent	
	Type	RPortPrototype	
	Interface	IF_ExchangedData	
	ComSpecs	QueuedReceiverComSpec for dataElement DE_ExchangedEvent, with queueLength = 2.	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_LowerTester	
	Requirements		
	VariableDataPrototype	**Name**	VDP_BusWrite
		Type	DataSendPoint OR DataWriteAccess
		Access to	PP_BusSendData/ IF_ExchangedData/ DE_ExchangedData
		Requirements	
		Name	VDP_BusRead
		Type	DataReceivePoint OR DataReadAccess
		Access to	RP_BusSendData/ IF_ExchangedData/ DE_ExchangedData
		Requirements	
		Name	VDP_BusSend
		Type	DataSendPoint
		Access to	PP_BusSendEvent/ IF_ExchangedData/ DE_ExchangedEvent
		Requirements	

		Name	VDP_BusReceive
		Type	DataReceivePoint
		Access to	RP_BusReceiveEvent/ IF_ExchangedData/ DE_ExchangedEvent
		Requirements	

6.1.2.4 Mandatory vs. Customizable Parts

In the configuration set, the definition of the queuing attribute of the port must not be changed.

However, the definition of the type of TYP_Exchanged data along with the init values for all ports as well as the values used in the test cases can be adjusted to the user needs.

6.1.3 Test Case Data Types

Test cases are configurable in order to better match requirements from users. They are abstracted from the data types and the values to be sent.
Different data sets can be used to better match user coverage needs. Data sets are defined using equivalence classes method to have a better coverage, as for example, data sets to transmit 8 bits data, 64 bits data, X bits data where X > maximum length of data on the bus (*e.g.* >8 bytes for CAN)…

List of parameters used by the test cases

Name	Type	Definition	Requirements
Value_Send1	TYP_ExchangedData	First data to transmit.	All data are different.
Value_Send2	TYP_ExchangedData	Second data to transmit.	
Value_Send3	TYP_ExchangedData	Third data to transmit.	
Value_Init_Send	TYP_ExchangedData	Initial value of Control SWC's port	
Value_Init_Receive1	TYP_ExchangedData	Initial value of observation 1 SWC's port	
Value_Init_Receive2	TYP_ExchangedData	Initial value of observation 2 SWC's port	
Value_Init_BusSend	TYP_ExchangedData	Initial value of Lower Tester's transmission port	
Value_Init_BusReceive	TYP_ExchangedData	Initial value of Lower Tester's reception port	

Timing_MsgOnBus	*<User choice>*	Period during which an actor waits between two actions.	Must be greater than the time require for a message to appear at least once on the bus.

Examples of test case data sets:

Test Case Data Set 1

Parameter	Value
TYP_ExchangedData	Unsigned integer 8 bits
Value_Send1	0x5A
Value_Send2	0x96
Value_Send3	0x56
Value_Init_Send	0x7B
Value_Init_Receive1	0xBD
Value_Init_Receive2	0xEB
Value_Init_BusSend	0xE7
Value_Init_BusReceive	0x7D

Test Case Data Set 2

Parameter	Value
TYP_ExchangedData	Float 64
Value_Send1	0x0569A65A
Value_Send2	0x2A965A96
Value_Send3	0xA956A956
Value_Init_Send	0xB7BDED7B
Value_Init_Receive1	0x9EDB7EBD
Value_Init_Receive2	0x3BE7D7EB
Value_Init_BusSend	0x1E7BDBE7
Value_Init_BusReceive	0xBEDB7E7D

Test Case Data Set 3 – For testing values greater than size limit of CAN messages

Parameter	Value
TYP_ExchangedData	Array Implementation Data Type : unsigned integer 8bits[9]
Value_Send1	0x5A965695A
Value_Send2	0x59A965A96
Value_Send3	0x9A956A956
Value_Init_Send	0xEB7BDED7B
Value_Init_Receive1	0x79EDB7EBD
Value_Init_Receive2	0x3BE73D7EB
Value_Init_BusSend	0xDE7EBDBE7
Value_Init_BusReceive	0xBEDBD7E7D

6.2 Re-usable Test Steps

None.

6.3 Test Cases

6.3.1 [ATS_RTE_00009] Implicit write and read / data – intra-ECU

Test Objective	Implicit write and read / data – intra-ECU		
ID	ATS_RTE_00009	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022 ATR: ATR_ATR_00023		
Trace to SWS Item	RTE: SWS_Rte_03571 RTE: SWS_Rte_03573 RTE: SWS_Rte_03744		
Requirements / Reference to Test Environment	ControlSWC and Observation1SWC are required. The ports ControlSWC.PP_SendData and Observation1SWC.RP_ReceiveData1 are connected.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Value_Send3 I-PDU Transmission Mode Timing: EventControlledTiming Implicit read access and implicit write access do not belong to the same coherency group. Hint: The TCPs in Main3 and Main5 require the OS and RTE to be configured such that RUN_Obs1 can preempt RUN_Ctrl. This means in particular that the OS task executing the runnable entity ControlSWC.RUN_Ctrl shall be preemptable. The TCP can be implemented for example with a pair of waitpoints (e.g. RUN_Ctrl send an event which activates RUN_Obs1, and waits with a blocking Rte_Receive that RUN_Obs1 produce another event).		
Summary	Verify that data element with “data” semantics (not queued) written by a sender SWC in an implicit way is read by another receiver SWC running on the same ECU in an implicit way. Ensure implicit semantics is correctly implemented. **Test Description:** Control starts, does an implicit write on its provide port then its execution is suspended and Observation starts. Observation starts and reads its require port. Data read is the init value and not the one written by Control. Observation terminates.		

	Control resumes, performs two implicit writes and terminates. Observation starts and reads the last data written by Observation.	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_ReceiveData1 during the execution of the test case	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Ctrl* starts	
Step 2	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite* with value *Value_Send1.* [SWS_Rte_03744]	
Step 3	**[CP]** *RUN_Ctrl* is suspended. [SWS_Rte_03571] *RUN_Obs1* starts	
Step 4	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes *Rte_IRead* on *VDP_ImplRead1*.	**[RUN<RUN_Obs1>]** Read data is initial value *Value_Init_Receive1*. [SWS_Rte_03571]
Step 5	**[CP]** *RUN_Obs1* terminates. *RUN_Ctrl* is resumed	
Step 6	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite* with value *Value_Send2*. [SWS_Rte_03573]	
Step 7	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite* with value *Value_Send3*. [SWS_Rte_03573]	
Step 8	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates. [SWS_Rte_03571]	
Step 9	**[CP]** *RUN_Obs1* starts.	
Step 10	**[RUN<RUN_Obs1>]**	**[RUN<RUN_Obs1>]**

	RUN_Obs1 executes *Rte_IRead* from *VDP_ImplRead1*.	Read data is *Value_Send3*. [SWS_Rte_03571], [SWS_Rte_03573]
Step 11	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Post-conditions	Run_Ctrl and Run_Obs are terminated	

6.3.2 [ATS_RTE_00010] Explicit write and read / data – intra-ECU

Test Objective	Explicit write and read / data – intra-ECU		
ID	ATS_RTE_00010	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022 ATR: ATR_ATR_00023		
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01091 RTE: SWS_Rte_02635 RTE: SWS_Rte_06011 RTE: SWS_Rte_06016		
Requirements / Reference to Test Environment	Required SWC: ControlSWC and Observation1SWC. Test Environment architecture: (see 1:1 intra-ECU test cases) port ControlSWC.PP_SendData connected to port Observation1SWC.RP_ReceiveData1.		
Configuration Parameters	Parameters used: • Value_Send1 • Value_Send2 I-PDU Transmission Mode Timing: EventControlledTiming Hint: The TCPs in Main3 and Main5 require the OS and RTE to be configured such that RUN_Obs1 can preempt RUN_Ctrl. This means in particular that the OS task executing the runnable entity ControlSWC.RUN_Ctrl shall be preemptable. The TCP can be implemented for example with a pair of waitpoints (e.g. RUN_Ctrl send an event which activates RUN_Obs1, and waits with a blocking Rte_Receive that RUN_Obs1 produce another event).		
Summary	Verify that data element with “data” semantics (not queued) written by a sender SWC in an explicit way is read by another receiver SWC running on the same ECU in an explicit way. Ensure explicit semantics is correctly implemented. **Test description:** Control starts, does an explicit write with Value_Send1 on the provided port then its execution is suspended and Observation starts.		

	Observation reads its required port. Data read is not the init value but the one written by Control. Observation terminates. Control resumes, performs another explicit write with Value_Send2 and terminates Observation starts and reads the last data written by Observation. Data read is the new value written by Control. Observation terminates.	
Needed Adaptation to other Releases		
Pre-conditions	Pre 1: The ControlSWC runnable contains the corresponding DataSendPoint. Pre 2: No other runnables than RUN_Ctrl performs a write to RP_ReceiveData1 during the execution of the test case. Pre 3: RTE_Read call is non-blocking (i.e. Observation1SWC contains DataReceivePoint referencing the DataElementPrototype for which the API is being generated, but no WaitPoint referencing the DataReceivePoint),	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Ctrl* starts.	
Step 2	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite* with value *Value_Send1.* [SWS_Rte_01071], [SWS_Rte_06016]	
Step 3	**[CP]** *RUN_Ctrl* is suspended after Rte_Write [SWS_Rte_02635] *RUN_Obs1* starts.	
Step 4	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes *Rte_Read* on *VDP_ExplRead1*. [SWS_Rte_01091], [SWS_Rte_06011]	**[RUN<RUN_Obs1>]** Read data is the value *Value_Send1*.
Step 5	**[CP]** *RUN_Obs1* terminates. *RUN_Ctrl* is resumed	
Step 6	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite* with value *Value_Send2*. [SWS_Rte_01071], [WS_Rte_06016]	
Step 7	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates.	
Step 8	**[CP]** *RUN_Obs1* starts.	

Step 9	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes *Rte_Read* on *VDP_ExplRead1*. [SWS_Rte_01091], [SWS_Rte_06011]	**[RUN<RUN_Obs1>]** Read data is the value *Value_Send2*.
Step 10	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Post-conditions	Run_Ctrl and Run_Obs terminated	

6.3.3 [ATS_RTE_00011] Send and receive / event – intra-ECU

Test Objective	Send and receive / event – intra-ECU		
ID	ATS_RTE_00011	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022 ATR: ATR_ATR_00023		
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_01092 RTE: SWS_Rte_02633		
Requirements / Reference to Test Environment	Required SWC: ControlSWC and Observation1SWC. Test Environment architecture: (see 1:1 intra-ECU test cases) port ControlSWC.PP_SendEvent connected to port Observation1SWC.RP_ReceiveEvent1.		
Configuration Parameters	Parameters used: • Value_Send1 • Value_Send2 • Value_Send3 I-PDU Transmission Mode Timing: EventControlledTiming		
Summary	Verify that two different data elements with “event” semantics (queued) sent one after the other by a sender SWC are received in the same order by another receiver SWC running on the same ECU. **Test Desciption** Control starts, sends three events then it terminates. The third event will be discarded as queue is full (queue length set to 2). Observation starts and asks for the reception of a first event from its required port. Event value is the first one send by Control. Observation asks for the reception of a second event from its required port. Event value is the second one sent by Control. Observation asks for the reception of a third event from its required port. No more events have been received.		

Needed Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Ctrl* starts.	
Step 2	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Send on *VDP_Send* with value *Value_Send1.* [SWS_Rte_01072], [SWS_Rte_02633]	
Step 3	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Send on *VDP_Send* with value *Value_Send2.* [SWS_Rte_01072], [SWS_Rte_02633]	
Step 4	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Send on *VDP_Send* with value *Value_Send3.* [SWS_Rte_01072], [SWS_Rte_02633]	**[RUN<RUN_Ctrl>]** RTE_Send returns RTE_E_LIMIT, the event Value_send3 has been discarded as queue is full (queue length set to 2).
Step 5	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates	
Step 6	**[CP]** *RUN_Obs1* is started	
Step 7	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*. [SWS_Rte_01092]	**[RUN<RUN_Obs1>]** Event read is *Value_Send1.*
Step 8	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*. [SWS_Rte_01092]	**[RUN<RUN_Obs1>]** Event read is *Value_Send2.*
Step 9	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*. [SWS_Rte_01092]	**[RUN<RUN_Obs1>]** Status returned is RTE_E_NO_DATA.
Step 10	**[RUN<RUN_Obs1>]**	

	RUN_Obs1 terminates	
Post-conditions	Run_Ctrl and Run_Obs terminated	

6.3.4 [ATS_RTE_00012] Implicit write / data – inter-ECU

Test Objective	Implicit write / data – inter-ECU		
ID	ATS_RTE_00012	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_03571 RTE: SWS_Rte_03573		
Requirements / Reference to Test Environment	ControlSWC and LowerTesterSWC are required. The ports ControlSWC.PP_SendData and LowerTesterSWC.RP_BusReceiveData are connected. Bus tool to emulate ECU executing LowerTesterSWC.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - CyclicTiming (cycle time shorter than Timing_MsgOnBus) Implicit read access and implicit write access do not belong to the same coherency group.		
Summary	Verify that a data element with “data” semantic (not queued) is transmitted on the bus if an implicit write is realized by a sender SWC connected to another receiver SWC running on a different ECU. **Test Description** Lower Tester is started and checks the bus. Control starts, does an implicit write on its provide port and waits for a given amount of time. During this time, Lower Tester must not detect any message from Control with the written data. Control does a second implicit write with a different value and waits for a given amount of time. During this time, Lower Tester must not detect any message from Control containing the latest written data. Control terminates and Lower Tester must detect a message from Control containing the latest written data.		
Needed Adaptation to other Releases			

Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_BusReceiveData during the execution of the test case.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Lower Tester starts.	
Step 2	**[CP]** *RUN_Ctrl* starts.	
Step 3	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite* with value *Value_Send1.* [SWS_Rte_03573]	
Step 4	**[RUN<RUN_Ctrl>]** WAIT for *Timing_MsgOnBus* ms	**[LT]** WHILE WAITING, no message from *RUN_Ctrl* with data *Value_Send1* found on the bus.
Step 5	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite* with value *Value_Send2.* [SWS_Rte_03573]	
Step 6	**[LT]** *Lower Tester* reads the bus for a message from *RUN_Ctrl* containing value *Value_Send2*.	
Step 7	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates.	
Step 8	**[CP]** WAIT for *Timing_MsgOnBus* ms.	
Step 9		**[LT]** A message from *RUN_Ctrl* containing value *Value_Send2* was available on the bus. [SWS_Rte_03571], [SWS_Rte_03573]
Post-conditions	RUN_Ctrl is terminated	

6.3.5 [ATS_RTE_00013] Explicit write / data – inter-ECU

Test Objective	Explicit write / data – inter-ECU

ID	ATS_RTE_00013	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_01071		
Requirements / Reference to Test Environment	ControlSWC and LowerTesterSWC are required. The ports ControlSWC.PP_SendData and LowerTesterSWC.RP_BusReceiveData are connected. Bus tool to emulate ECU executing LowerTesterSWC.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus)		
Summary	Verify that a data element with “data” semantics (not queued) is transmitted on the bus if an explicit write is realized by a sender SWC connected to another receiver SWC running on a different ECU. **Test Description** Lower Tester is started and checks the bus. Control starts, does an explicit write on its provide port and waits for a given amount of time. During this time, Lower Tester must detect a message from Control containing the latest written data. Control does a second explicit write with a different value and terminates. Lower Tester must detect a message from Control containing the latest written data		
Needed Adaptation to other Releases			
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_BusReceiveData during the execution of the test case.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Lower Tester starts.		
Step 2	**[CP]** *RUN_Ctrl* starts.		
Step 3	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite* with value *Value_Send1.*		

Step 4	**[RUN<RUN_Ctrl>]** WAIT for *Timing_MsgOnBus* ms.	**[LT]** WHILE WAITING, a message from *RUN_Ctrl* containing value *Value_Send1* is available on the bus.
Step 5	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite* with value *Value_Send2.* [SWS_Rte_01071]	
Step 6	**[LT]** *Lower Tester* reads the bus for a message from *RUN_Ctrl* containing value *Value_Send2*.	
Step 7	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates.	
Step 8	**[CP]** WAIT for *Timing_MsgOnBus* ms.	
Step 9		**[LT]** A message from *RUN_Ctrl* containing value *Value_Send2* was available on the bus.
Post-conditions	RUN_Ctrl is terminated	

6.3.6 [ATS_RTE_00014] Send / events – inter-ECU

Test Objective	Send / events – inter-ECU		
ID	ATS_RTE_00014	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_07827		
Requirements / Reference to Test Environment	ControlSWC and LowerTesterSWC are required. The ports ControlSWC.PP_SendEvent and LowerTesterSWC.RP_BusReceiveEvent are connected. Bus tool to emulate ECU executing LowerTesterSWC.		
Configuration Parameters	Parameters: - Value_Send1 - Timing_MsgOnBus		

	I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus)	
Summary	Verify that a data element with “event” semantic (queued) is transmitted on the bus if a send is realized by a sender SWC connected to another receiver SWC running on a different ECU **Test Description** Control starts, performs a send of an event then terminates. Lower Tester must detect a message from Control containing the latest written event.	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_BusReceiveEvent during the execution of the test case.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Lower Tester starts	
Step 2	**[CP]** *RUN_Ctrl* starts	
Step 3	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_Send on *VDP_Send* with value *Value_Send1.*	
Step 4	**[LT]** *Lower Tester* reads the bus for a message from *RUN_Ctrl* containing value *Value_Send1*.	
Step 5	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates.	
Step 6	**[CP]** WAIT for *Timing_MsgOnBus* ms.	
Step 7		**[LT]** A message from *RUN_Ctrl* containing value *Value_Send1* was available on the bus.
Post-conditions	RUN_Ctrl is terminated	

6.3.7 [ATS_RTE_00015] Implicit read / data – inter-ECU

Test Objective	Implicit read / data – inter-ECU

ID	ATS_RTE_00015	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment	Observation1SWC and LowerTesterSWC are required. The ports Observation1SWC.RP_ReceiveData1 and LowerTesterSWC.PP_BusSendData are connected. Bus tool to emulate ECU executing sender SWC.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Value_Send3 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus) Implicit read access and implicit write access do not belong to the same coherency group.		
Summary	Verify that a data element with “data” semantic transmitted via the bus is read by one receiver SWC performing implicit reception if the receiver SWC is connected to another sender SWC running on a different ECU. Ensure implicit semantic is correctly implemented. **Test Description** Observation starts and waits for a given amount of time. During this time Lower Tester send a message on the bus to Observation containing a first data. After the wait, Observation performs an implicit read. Data read is init value. Observation then terminates.		
Needed Adaptation to other Releases			
Pre-conditions	No other runnables than RUN_LowerTester performs a write to RP_ReceiveData1 during the execution of the test case.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** *RUN_Obs1* starts.		
Step 2	**[RUN<RUN_Obs1>]**		

	WAIT till *Lower Tester* performs step Main 3. WHILE WAITING, DO nothing	
Step 3	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send1*.	
Step 4	**[RUN<RUN_Obs1>]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 5	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_ImplRead1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Init_Receive1*.
Step 6	**[RUN<RUN_Obs1>]** RUN_Obs1 terminates	
Step 7	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send2*.	
Step 8	**[LT]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 9	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send3*.	
Step 10	**[CP]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 11	**[CP]** *RUN_Obs1* starts.	
Step 12	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_ImplRead1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Send3*.
Step 13	**[RUN<RUN_Obs1>]** RUN_Obs1 terminates.	
Post-conditions	RUN_Obs1 is terminated	

6.3.8 [ATS_RTE_00016] Explicit read / data – inter-ECU

Test Objective	Explicit read / data – inter-ECU		
ID	ATS_RTE_00016	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_01091		
Requirements / Reference to Test Environment	Observation1SWC and LowerTesterSWC are required. The ports Observation1SWC.RP_ReceiveData1 and LowerTesterSWC.PP_BusSendData are connected. Bus tool to emulate ECU executing sender SWC.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Value_Send3 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus)		
Summary	Verify that a data element with “data” semantic transmitted via the bus is read by one receiver SWC performing explicit reception if the receiver SWC is connected to another sender SWC running on a different ECU. Ensure explicit semantic is correctly implemented. **Test Description** Observation starts and waits for Lower Tester to send a message on the bus to Observation containing a first data. After the wait, Observation performs an explicit read. Data read is the latest value send by Lower tester. Observation then terminates. Observation starts, does an explicit read on its require port. Data read is the latest value send by Lower tester. Observation waits for Lower Tester to send two messages on the bus to Observation containing two different data. After the wait, Observation performs an explicit read. Data read is the latest value send by Lower tester.		
Needed Adaptation to other Releases			

Pre-conditions	No other runnables than RUN_LowerTester performs a write to RP_ReceiveData1 during the execution of the test case.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Obs1* starts.	
Step 2	**[RUN<RUN_Obs1>]** WAIT till *Lower Tester* performs step Main 3. WHILE WAITING, DO nothing	
Step 3	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send1*.	
Step 4	**[RUN<RUN_Obs1>]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 5	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Read on *VDP_ExplRead1*	**[RUN<RUN_Obs1>]** Data read is *Value_Send1*
Step 6	**[RUN<RUN_Obs1>]** RUN_Obs1 terminates	
Step 7	**[CP]** *RUN_Obs1* starts.	
Step 8	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Read on *VDP_ExplRead1*	**[RUN<RUN_Obs1>]** Data read is *Value_Send1.*
Step 9	**[RUN<RUN_Obs1>]** WAIT till Lower Tester performs step Main 12. WHILE WAITING, DO nothing	
Step 10	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send2*.	
Step 11	**[LT]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 12	**[LT]**	

	Lower Tester sends on the bus a message to *RP_ReceiveData1* with *Value_Send3*.	
Step 13	**[RUN<RUN_Obs1>]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 14	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Read on *VDP_ExplRead1*	**[RUN<RUN_Obs1>]** Data read is *Value_Send3*.
Step 15	**[RUN<RUN_Obs1>]** RUN_Obs1 terminates.	
Post-conditions	RUN_Obs1 is terminated	

6.3.9 [ATS_RTE_00017] Receive / events – inter-ECU

Test Objective	Receive / events – inter-ECU		
ID	ATS_RTE_00017	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020		
Trace to SWS Item	RTE: SWS_Rte_01092		
Requirements / Reference to Test Environment	Observation1SWC and LowerTesterSWC are required. The ports Observation1SWC.RP_ReceiveEvent1 and LowerTesterSWC.PP_BusSendEvent are connected. Bus tool to emulate ECU executing sender SWC.		
Configuration Parameters	Parameters: - Value_Send1 - Value_Send2 - Value_Send3 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus)		
Summary	Verify that two data elements with “event” semantic (queued) transmitted on the bus one after another are read in the same order by a receiver SWC connected to another sender SWC running on a different ECU. **Test Description:**		

	Lower Tester sends three messages on the bus to Observation with different event values. The third event will be discarded as queue is full (queue length set to 2). Observation starts and asks for the reception of a first event from its require port. Event value is the first one send by Lower Tester. Observation asks for the reception of a second event from its require port. Event value is the second one send by Lower Tester. Observation asks for the reception of a third event from its require port. No more event has been received.	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_LowerTester performs a write to RP_ReceiveEvent1 during the execution of the test case	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveEvent1* with *Value_Send1*.	
Step 2	**[LT]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 3	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveEvent1* with *Value_Send2*.	
Step 4	**[LT]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 5	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveEvent1* with *Value_Send3*.	
Step 6	**[LT]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 7	**[CP]** *RUN_Obs1* is started	
Step 8	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Send1.*

Step 9	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Send2.*
Step 10	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_Receive on *VDP_Receive1*.	**[RUN<RUN_Obs1>]** Status returned is RTE_E_NO_DATA
Step 11	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Post-conditions	RUN_Obs1 is terminated	

6.3.10 [ATS_RTE_00018] 1:n reception from inter-ECU

Test Objective	1:n reception from inter-ECU		
ID	ATS_RTE_00018	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022		
Trace to SWS Item			
Requirements / Reference to Test Environment	Observation1SWC, Observation2SWC and LowerTesterSWC are required. The ports Observation1SWC.RP_ReceiveData1 and LowerTesterSWC.PP_BusSendData are connected. The ports Observation2SWC.RP_ReceiveData2 and LowerTesterSWC.PP_BusSendData are connected. Bus tool to emulate ECU executing sender SWC.		
Configuration Parameters	Parameters: - Value_Send1 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming - CyclicTiming (cycle time shorter than Timing_MsgOnBus)		
Summary	Ensure that a data received from the bus is read by two different receiver SWCs running on the same ECU and connected to the same sender SWC running on another ECU. **Test Description:** Lower Tester to send a message on the bus to Observations containing a first data. Both Observations start again and perform a read on their require port. Data read are the value send by Lower Tester.		

	It is recommended that Observations use a different read method, *ie*. one uses explicit read and the other one implicit read.	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_LowerTester performs a write to RP_ReceiveData1 during the execution of the test case	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT]** *Lower Tester* sends on the bus a message to *RP_ReceiveData1* with *Value_Send1*.	
Step 2	**[CP]** WAIT for *Timing_MsgOnBus* ms. WHILE WAITING, DO nothing	
Step 3	**[CP]** *RUN_Obs1* is started	
Step 4	**[RUN<RUN_Obs1>]** RUN_Obs1 reads its port RP_ReceiveData1. (Implementer choice whether read is explicit or implicit: *RUN_Obs1* executes Rte_Read on *VDP_ExplRead1*. OR *RUN_Obs1* executes Rte_IRead on *VDP_implRead1*.)	**[RUN<RUN_Obs1>]** Data read is *Value_Send1*
Step 5	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Step 6	**[CP]** *RUN_Obs2_1* is started	
Step 7	**[RUN<RUN_Obs2_1>]** RUN_Obs2.1 reads its port RP_ReceiveData2.	**[RUN<RUN_Obs2_1>]** Data read is *Value_Send1*

	(Implementer choice whether read is explicit or implicit: *RUN_Obs2_1* executes Rte_Read on *VDP_ExplRead2_1*. OR *RUN_Obs2_1* executes Rte_IRead on *VDP_ImplRead2_1*.)	
Step 8	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* terminates	
Post-conditions	Run_Obs1 and RUN_Obs2.1 are terminated	

6.3.11 [ATS_RTE_00019] 1:n transmission to intra-ECU and inter-ECU

Test Objective	1:n transmission to intra-ECU and inter-ECU		
ID	ATS_RTE_00019	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022		
Trace to SWS Item			
Requirements / Reference to Test Environment	ControlSWC, Observation1SWC and LowerTesterSWC are required. The ports Observation1SWC.RP_ReceiveData1 and ControlSWC.PP_SendData are connected. The ports LowerTesterSWC.PP_BusReceiveData and ControlSWC.PP_SendData are connected. Bus tool to emulate ECU executing sender SWC.		
Configuration Parameters	Parameters: - Value_Send1 - Timing_MsgOnBus I-PDU Transmission Mode Timing: - EventControlledTiming		
Summary	Ensure that a data written by a sender SWC connected to two different receiver SWCs respectively running on the same ECU and on another ECU is correctly transmitted to all receiver SWCs. **Test Description** Control starts and performs a write on its provide port then terminates.		

	Observation start, performs a read on its require port. Data read is the value send by Control. Lower Tester must detect a message from Control containing the written data	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_ReceiveData1 and RP_BusReceiveData during the execution of the test case	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Ctrl* starts.	
Step 2	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* writes its port P*P_SendData* value *Value_Send1.* (Implementer choice whether write is explicit or implicit: *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite*. OR *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite*.)	
Step 3	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates	
Step 4	**[CP]** *RUN_Obs1* is started	
Step 5	**[RUN<RUN_Obs1>]** *RUN_Obs1* reads its port RP_ReceiveData1. (Implementer choice whether read is explicit or implicit: *RUN_Obs1* executes *Rte_Read* on *VDP_ExplRead1*. OR	**[RUN<RUN_Obs1>]** Data read is *Value_Send1*

	RUN_Obs1 executes *Rte_IRead* on *VDP_implRead1*.)	
Step 6	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Step 7	**[CP]** WAIT for *Timing_MsgOnBus* ms	**[LT]** WHILE WAITING, a message from *RUN_Ctrl* containing value *Value_Send1* is available on the bus.
Post-conditions	RUN_Ctrl and RUN_Obs1 are terminated.	

6.3.12 [ATS_RTE_00020] Init value, Never received status, and valid 1 to 2 write

Test Objective	Init value, Never received status, and valid 1 to 2 write		
ID	ATS_RTE_00020	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022 ATR: ATR_ATR_00026		
Trace to SWS Item	RTE: SWS_Rte_06010 RTE: SWS_Rte_07644		
Requirements / Reference to Test Environment	ControlSWC, Observation1SWC and Observation2SWC are required. The ports Observation1SWC.RP_ReceiveData1 and ControlSWC.PP_SendData are connected. The ports Observation2SWC.RP_ReceiveData2 and ControlSWC.PP_SendData are connected.		
Configuration Parameters	Parameters: - Value_Send1 I-PDU Transmission Mode Timing: - EventControlledTiming		
Summary	Verify that when a valid init value is configured and no data have been transmitted then two receiver SWCs connected to the same sender SWC, performing respectively an implicit reception and an explicit reception, will read the init value and get the status NEVER_RECEIVED. Then when the sender SWC performs a write with a valid data, two receiver SWCs connected to the sender SWC, performing respectively an implicit reception and an explicit reception, will read the transmitted data. **Test Description** Both Observations start, perform a read and their require ports. Data read are the initial values. Both Observations then terminate.		

	Control starts, does a write on its provide port then terminates. Both Observations start again and perform a read on their require port. Data read are the value send by Control.	
Needed Adaptation to other Releases		
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_ReceiveData1 and RP_ ReceiveData2 during the execution of the test case	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Obs1* starts	
Step 2	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_ImplRead1*	**[RUN<RUN_Obs1>]** Data read is *Value_Init_Receive1*
Step 3	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IStatus on *VDP_ImplRead1*	**[RUN<RUN_Obs1>]** Status returned is *RTE_E_NEVER_RECEIVED*
Step 4	**[RUN<RUN_Obs1>]** *Run_Obs1* terminates	
Step 5	**[CP]** *RUN_Obs2_1* starts	
Step 6	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* executes Rte_Read on *VDP_ExplRead2_1*.	**[RUN<RUN_Obs2_1>]** Data read is *Value_Init_Receive2*. Status returned is *RTE_E_NEVER_RECEIVED*
Step 7	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* terminates	
Step 8	**[CP]** *RUN_Ctrl* starts	
Step 9	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* writes its port P*P_SendData* value *Value_Send1.* (Implementer choice whether write is explicit or implicit: *RUN_Ctrl* executes Rte_Write on *VDP_ExplWrite*.	

	OR *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite*.)	
Step 10	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates	
Step 11	**[CP]** *RUN_Obs1* starts	
Step 12	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_ImplRead1*	**[RUN<RUN_Obs1>]** Data read is *Value_Send1*
Step 13	**[RUN<RUN_Obs1>]** *Run_Obs1* terminates	
Step 14	**[CP]** *RUN_Obs2_1* starts.	
Step 15	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* executes Rte_Read on *VDP_ExplRead2_1*.	**[RUN<RUN_Obs2_1>]** Data read is *Value_Send1*
Step 16	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* terminates	
Post-conditions	Run_Ctrl, Run_Obs1 and RUN_Obs2.1 are terminated	

6.3.13 [ATS_RTE_00021] Implicit write and read in coherency group

Test Objective	Implicit write and read in coherency group		
ID	ATS_RTE_00021	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00020 ATR: ATR_ATR_00022		
Trace to SWS Item	RTE: SWS_Rte_07063 RTE: SWS_Rte_07066		
Requirements / Reference to Test Environment	ControlSWC, Observation1SWC and Observation2SWC are required. The ports Observation1SWC. RP_ReceiveData1 and ControlSWC.RUN_Ctrl.PP_SendData are connected. The ports Observation2SWC.RP_ReceiveData2 and ControlSWC.RUN_Ctrl.PP_SendData are connected.		

	ControlSWC.Run_Ctrl, Observation2SWC.Run_Obs2.1 and Observation2SWC.Run_Obs2.2 are executed in the same OS task. Order of execution in the OS task is Observation2SWC.Run_Obs2.1, then ControlSWC.Run_Ctrl, then Observation2SWC.Run_Obs2.2. Their DataWriteAccess and DataReadAccess belong to the same coherency group. Observation1WSC.Run_Obs1 does not belong to previous coherency group. Hint: The sequence of the test steps require the OS and RTE to be configured such that RUN_Obs1 can preempt RUN_Ctrl. This means in particular that the OS task executing the runnable entity ControlSWC.RUN_Ctrl shall be preemptable. The TCP can be implemented for example with a pair of waitpoints (e.g. RUN_Ctrl send an event which activates RUN_Obs1, an
Configuration Parameters	Parameters: - Value_Send1 I-PDU Transmission Mode Timing: - EventControlledTiming
Summary	Ensure coherency group are correctly supported by implicit communication **Test description** Observation 1 starts, performs a read on its require port. Data read is initial value. Observation 1 terminates. First runnable entity of Observation 2 starts, performs a read on its require port. Data read is initial value. Runnable entity terminates. Control starts, performs a write on its provide port then terminates. Observation 1 starts again, performs a read on its require port. Data read is value written by Control. Observation 1 terminates. The task executing the coherency group is still running. Second runnable entity of Observation 2 starts, performs a read on its require port. Data read is initial value. Runnable entity terminates. The task executing the coherency group is not terminated. First runnable entity of Observation 2 starts again, performs a read on its require port. Data read is value written by Control. Runnable entity terminates. Second runnable entity of Observation 2 starts again, performs a read on its require port. Data read is value written by Control. Runnable entity terminates.
Needed Adaptation to other Releases	
Pre-conditions	No other runnables than RUN_Ctrl performs a write to RP_ReceiveData1 and RP_ReceiveData2 during the execution of the test case

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** *RUN_Obs1* starts.	

Step 2	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_ExplRead1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Init_Receive1*
Step 3	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Step 4	**[CP]** *RUN_Obs2_1* starts	
Step 5	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* executes Rte_IRead on *VDP_implRead2.1*.	**[RUN<RUN_Obs2_1>]** Data read is *Value_Init_Receive2*. [SWS_Rte_07063]
Step 6	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* terminates	
Step 7	**[CP]** *RUN_Ctrl* starts.	
Step 8	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* executes Rte_IWrite on *VDP_ImplWrite*.	
Step 9	**[RUN<RUN_Ctrl>]** *RUN_Ctrl* terminates	
Step 10	**[CP]** *RUN_Obs1* starts	
Step 11	**[RUN<RUN_Obs1>]** *RUN_Obs1* executes Rte_IRead on *VDP_implRead1*.	**[RUN<RUN_Obs1>]** Data read is *Value_Send1*.
Step 12	**[RUN<RUN_Obs1>]** *RUN_Obs1* terminates	
Step 13	**[CP]** *RUN_Obs2_2* starts	
Step 14	**[RUN<RUN_Obs2_2>]** *RUN_Obs2_2* executes Rte_IRead on *VDP_ImplRead2_2*.	**[RUN<RUN_Obs2_2>]** Data read is *Value_Init_Receive2*. [SWS_Rte_07063], [SWS_Rte_07066]
Step 15	**[RUN<RUN_Obs2_2>]** *RUN_Obs2_2* terminates	
Step 16	**[CP]** *RUN_Obs2_1* starts	

Step 17	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* executes Rte_IRead on *VDP_ImplRead2._*	**[RUN<RUN_Obs2_1>]** Data read is *Value_Send1*
Step 18	**[RUN<RUN_Obs2_1>]** *RUN_Obs2_1* terminates	
Step 19	**[CP]** *RUN_Obs2_2* starts	
Step 20	**[RUN<RUN_Obs2_2>]** *RUN_Obs2.2* reads its port *RP_ReceiveData2*. (Implementer choice whether read is explicit or implicit: *RUN_Obs2_2* executes Rte_Read on *VDP_ExplRead2_2.* OR *RUN_Obs2_2* executes Rte_IRead on *VDP_ImplRead2_2*.)	**[RUN<RUN_Obs2_2>]** Data read is *Value_Send1*
Step 21	**[RUN<RUN_Obs2_2>]** *RUN_Obs2_2* terminates	
Post-conditions	None	

6.3.14 [ATS_RTE_00634] Explicit Nonqueued Inter-ECU Rte_Write For DataElement Of Primitive Data Type When Com Service Is Not Available

Test Objective	Explicit Nonqueued Inter-ECU Rte_Write For DataElement Of Primitive Data Type When Com Service Is Not Available		
ID	ATS_RTE_00634	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01280 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011		

	RTE: SWS_Rte_06023 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype to send - VariableDataPrototype_TC4 typed with a primitive data type with swImplPolicy 'standard' * a RunnableEntity to execute the test sequence with - with a dataSendPoint which references VariableDataPrototype_TC4 A communication matrix with * Signal_TC4 - mapped to VariableDataPrototype_TC4 - transmitted by the ECU * ISignalIPdu_TC4 - configured with an EventControlledTiming - Signal_TC4 mapped to ISignalIPdu_TC4 with transferProperty triggered * ISignalIPduGroup_TC4 which references ISignalIPdu_TC4
Summary	This test case verifies the explicit inter-ECU communication when the IPduGroup used to send a dataElement is not started. In a RunnableEntity Rte_Write is called which causes a transmission request (in case of inter-ECU communication by invoking the Com_SendSignal API). Since the IPduGroup is not started the Com_SendSignal API returns COM_SERVICE_NOT_AVAILABLE and in turn Rte_Write returns RTE_E_COM_STOPPED.
Needed Adaptation to other Releases	
Pre-conditions	ISignalIPduGroup_TC4 shall be in stopped mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write for VariableDataPrototype_TC4 with valid data.	[SWC] Rte_Write returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.15 [ATS_RTE_00635] Explicit Nonqueued Inter-ECU Rte_Write For DataElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Rte_Write For DataElement With Array Data Type		
ID	ATS_RTE_00635	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			

Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01280 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824	
Requirements / Reference to Test Environment		
Configuration Parameters	A SW-C with * a PPortPrototype to send - VariableDataPrototype_TC5array typed with an array data type with swImplPolicy 'standard' * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC5array A communication matrix with * SignalGroup_TC5array - mapped to VariableDataPrototype_TC5array - transmitted by the ECU * ISignalIPdu_TC5array - configured with an EventControlledTiming - SignalGroup_TC5array mapped to ISignalIPdu_TC5array with transferProperty triggered	
Summary	This test case verifies the explicit inter-ECU communication for a dataElement with array data type. Explicit inter-ECU sender-receiver communication shall be initiated to send valid data. Rte_Write will invoke Com_UpdateShadowSignal API for each element in the array and then it invokes Com_SendSignalGroup API to cause transmission. The frame is then observed on the bus.	
Needed Adaptation to other Releases		
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write for VariableDataPrototype_TC5array with valid data.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	-	[LT] The frame with ISignalIPdu_TC5array has been seen on the bus with the values of the different subElements of VariableDataPrototype_TC5array
Post-conditions	NONE	

6.3.16 [ATS_RTE_00636] Explicit Nonqueued Inter-ECU Rte_Write For DataElement With Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Rte_Write For DataElement With Record Data Type		
ID	ATS_RTE_00636	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01280 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype to send - VariableDataPrototype_TC5record typed with a record data type with swImplPolicy 'standard' * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC5record A communication matrix with * SignalGroup_TC5record - mapped to VariableDataPrototype_TC5record - transmitted by the ECU * ISignalIPdu_TC5record - configured with an EventControlledTiming - SignalGroup_TC5record mapped to ISignalIPdu_TC5record with transferProperty triggered		
Summary	This test case verifies the explicit inter-ECU communication for a dataElement with record data type. Explicit inter-ECU sender-receiver communication shall be initiated to send valid data. Rte_Write will invoke Com_UpdateShadowSignal API for each element in the record and then it invokes Com_SendSignalGroup API to cause transmission. The frame is then observed on the bus.		
Needed Adaptation to other Releases			
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write for VariableDataPrototype_TC5record with valid data.		[SWC] Rte_Write returns RTE_E_OK.
Step 2	-		[LT] The frame with ISignalIPdu_TC5 has

		been seen on the bus with the values of the different subElements of VariableDataPrototype_TC5record
Post-conditions	NONE	

6.3.17 [ATS_RTE_00637] Explicit Nonqueued Inter-ECU Rte_Read For DataElement Of Primitive Data Type

Test Objective	Explicit Nonqueued Inter-ECU Rte_Read For DataElement Of Primitive Data Type		
ID	ATS_RTE_00637	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype to receive - VariableDataPrototype_TC6 typed with a primitive data type with swImplPolicy ‘standard’ * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC6 - with a dataReceivePointByArgument which references VariableDataPrototype_TC6 A communication matrix with * Signal_TC6 - mapped to VariableDataPrototype_TC6 - received by the ECU		
Summary	This test case verifies the explicit nonqueued inter-ECU communication: a runnable can be invoked after reception of a signal and it can read the data received in the signal.		
Needed Adaptation to other Releases			
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[LT] Transmit a frame with Signal_TC6.		[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.

Step 2	[RUN<RunnableEntity_DRE>] Invoke the Rte_Read API to read VariableDataPrototype_TC6.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK. The data received has the same value as sent in the frame.
Post-conditions	NONE	

6.3.18 [ATS_RTE_00638] Explicit Nonqueued Inter-ECU Rte_Read For DataElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Rte_Read For DataElement With Array Data Type		
ID	ATS_RTE_00638	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SWC with * An RPortPrototype to receive - VariableDataPrototype_TC7array typed with an array data type with swImplPolicy ‘standard’ * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC7array - with a dataReceivePointByArgument which references VariableDataPrototype_TC7array A communication matrix with * SignalGroup_TC7array - mapped to VariableDataPrototype_TC7array - received by the ECU		
Summary	This test case verifies the explicit inter-ECU communication: a runnable can be invoked after reception of a signal and it can read the array data received from the bus.		
Needed Adaptation to other Releases			
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	[LT] Transmit frame withSignalGroup_TC7array.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke the Rte_Read API to read VariableDataPrototype_TC7array.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK. The data received has the same values as sent in the frame.
Post-conditions	NONE	

6.3.19 [ATS_RTE_00639] Explicit Nonqueued Inter-ECU Rte_Read For DataElement WIth Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Rte_Read For DataElement WIth Record Data Type		
ID	ATS_RTE_00639	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SWC with * An RPortPrototype to receive - VariableDataPrototype_TC7record typed with a record data type with swImplPolicy ‘standard’ * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC7record - with a dataReceivePointByArgument which references VariableDataPrototype_TC7record A communication matrix with * SignalGroup_TC7record - mapped to VariableDataPrototype_TC7record - received by the ECU		
Summary	This test case verifies the explicit inter-ECU communication: a runnable can be invoked after reception of a signal and it can read the record data received from the bus.		

Needed Adaptation to other Releases		
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit frame with SignalGroup_TC7record.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke the Rte_Read API to read VariableDataPrototype_TC7record.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK. The data received has the same values as sent in the frame.
Post-conditions	NONE	

6.3.20 [ATS_RTE_00640] Explicit Queued Inter-ECU Rte_Send For DataElement Of Variable-length Arrays Of Bytes

Test Objective	Explicit Queued Inter-ECU Rte_Send For DataElement Of Variable-length Arrays Of Bytes		
ID	ATS_RTE_00640	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_01281 RTE: SWS_Rte_04526 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011 RTE: SWS_Rte_07817 RTE: SWS_Rte_07821 RTE: SWS_Rte_07825		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype to send - VariableDataPrototype_TC8 typed by a dynamic length array (arraySizeSemantics set to variableSize maxNumberOfElements > 2) with swImplPolicy ‘queued’ * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC8 A communication matrix with * SignalGroup_TC8 - with dynamicLength = TRUE - mapped to VariableDataPrototype_TC8 * ISignalIPdu_TC8		

	- SignalGroup_TC8 mapped to ISignalIPdu_TC8 with transferProperty triggered - configured with an EventControlledTiming	
Summary	This test case verifies the explicit queued inter-ECU communication for variable-length array of bytes. The dataElement is sent with Rte_Send and it has to be transmitted on the bus with the right length.	
Needed Adaptation to other Releases		
Pre-conditions	ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Send on VariableDataPrototype_TC8 with an array of 2 bytes and the length set to 2.	[SWC] Rte_Send returns RTE_E_OK.
Step 2	-	[LT] The frame containing ISignalIPdu_TC8 is transmitted on the bus with the 2 bytes matching the values sent in step 1.
Post-conditions	NONE	

6.3.21 [ATS_RTE_00641] Explicit Queued Inter-ECU Rte_Receive For DataElement Of Variable-Length Arrays Of Bytes

Test Objective	Explicit Queued Inter-ECU Rte_Receive For DataElement Of Variable-Length Arrays Of Bytes		
ID	ATS_RTE_00641	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01092 RTE: SWS_Rte_01135 RTE: SWS_Rte_01288 RTE: SWS_Rte_02598 RTE: SWS_Rte_06011 RTE: SWS_Rte_07814 RTE: SWS_Rte_07817		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype to receive - VariableDataPrototype_TC9 typed by a dynamic length array (arraySizeSemantics set to variableSize maxNumberOfElements > 2) with swImplPolicy ‘queued’ * RunnableEntity_DRE		

	- started by a DataReceivedEvent on reception of VariableDataPrototype_TC9 - with a dataReceivePointByArgument which references VariableDataPrototype_TC9 A communication matrix with * SignalGroup_TC9 - mapped to VariableDataPrototype_TC9 - dynamicLength = TRUE
Summary	This test case verifies that the reception of data for an array of variable length activates an associated DRE runnable and the correct data is received by applications through the Rte_Receive API.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] transmit frame withSignal_TC9Rx.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke the Rte_Receive API to read variable length data received on VariableDataPrototype_TC9.	[RUN<RunnableEntity_DRE>] Rte_Receive returns RTE_E_OK. The data received has the same value and length as sent in the frame.
Post-conditions	NONE	

6.3.22 [ATS_RTE_00642] Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'HandleInvalid = Keep' And 'HandleNeverReceived = False' Or Undefined

Test Objective	Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'HandleInvalid = Keep' And 'HandleNeverReceived = False' Or Undefined		
ID	ATS_RTE_00642	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01282 RTE: SWS_Rte_01289 RTE: SWS_Rte_02626 RTE: SWS_Rte_05030 RTE: SWS_Rte_06011 RTE: SWS_Rte_07396 RTE: SWS_Rte_08005 RTE: SWS_Rte_08008		

Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC10 with the same value as initValue and invalidValue (Ex: 0xAA) with swImplPolicy 'standard' - is typed by a SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC10 (handleInvalid = keep) * An RPortPrototype to receive VariableDataPrototype_TC10 - NonqueuedReceiverComSpec with handleNeverReceived = FALSE - connected with the previous PPortPrototype * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC10 - with a dataSendPoint which references VariableDataPrototype_TC10		
Summary	If the initValue of a nonqueued dataElement equals the invalidValue and handleInvalid is set to keep and the handleNeverReceived is set to FALSE or not defined then data read shall be returned as invalid and should provide the invalidValue until the first reception of the dataElement. If a dataElement has been invalidated in case of Intra-partition communication and the attribute handleInvalid is set to keep – the query of the value shall return the invalid value.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC10.		[SWC] Rte_Read returns RTE_E_INVALID and provides the invalidValue.
Step 2	[SWC] Invoke Rte_Invalidate to invalidate the dataElement VariableDataPrototype_TC10.		[SWC] Rte_Invalidate returns RTE_E_OK indicating invalidation successful.
Step 3	[SWC] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC10.		[SWC] Rte_Read returns RTE_E_INVALID and provides the invalidValue.
Post-conditions	NONE		

6.3.23 [ATS_RTE_00643] Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'HandleInvalid = Keep' And 'HandleNeverReceived = True'

Test Objective	Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'HandleInvalid = Keep' And 'HandleNeverReceived = True'		
ID	ATS_RTE_00643	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01282 RTE: SWS_Rte_01289 RTE: SWS_Rte_02626 RTE: SWS_Rte_05030 RTE: SWS_Rte_06011 RTE: SWS_Rte_07382 RTE: SWS_Rte_07396 RTE: SWS_Rte_07643 RTE: SWS_Rte_08009
Requirements / Reference to Test Environment	
Configuration Parameters	A SWC with * a PPortPrototype to send - VariableDataPrototype_TC11 with the same value as initValue and invalidValue (Ex: 0xAA) with swImplPolicy ‘standard’ - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC11 (handleInvalid = keep) * An RPortPrototype to receive VariableDataPrototype_TC11 - NonqueuedReceiverComSpec with handleNeverReceived = TRUE - connected with the previous PPortPrototype * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC11 - with a dataSendPoint which references VariableDataPrototype_TC11
Summary	If the initValue of a nonqueued dataElement equals the invalidValue and handleInvalid is set to keep and the handleNeverReceived is set to TRUE the RTE API Rte_Read should return RTE_E_NEVER_RECEIVED until first reception of the dataElement. In this case Rte_Read shall provide the initValue. If a data element has been invalidated in case of intra-partition communication and the attribute handleInvalid is set to keep – the query of the value shall return the invalidValue.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Read API to explicitly read the dataElement VariableDataPrototype_TC11.	[SWC] Rte_Read returns RTE_E_NEVER_RECEIVED (indicating that no data has been received due to system start) and provides the initValue.
Step 2	[SWC] Invoke Rte_Invalidate API to invalidate the dataElement VariableDataPrototype_TC11.	[SWC] Rte_Invalidate returns RTE_E_OK indicating invalidation successful.

Step 3	[SWC] Invoke Rte_Read API to explicitly read the dataElement VariableDataPrototype_TC11.	[SWC] Rte_Read returns RTE_E_INVALID and provides the invalidValue.
Post-conditions	NONE	

6.3.24 [ATS_RTE_00644] Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'Handleinvalid = Replace'

Test Objective	Explicit Nonqueued Intra-Partition Data Invalidation When Attribute 'Handleinvalid = Replace'		
ID	ATS_RTE_00644	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01282 RTE: SWS_Rte_01289 RTE: SWS_Rte_05049 RTE: SWS_Rte_06011 RTE: SWS_Rte_07396		
Requirements / Reference to Test Environment			
Configuration Parameters	A SWC with * a PPortPrototype to send - VariableDataPrototype_TC12 with different values as initValue (Ex: 0x12) and invalidValue (Ex: 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC12 (handleInvalid = replace) * an RPortPrototype to receive VariableDataPrototype_TC12 - NonqueuedReceiverComSpec with handleNeverReceived = FALSE - connected with the previous PPortPrototype * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC12 - with a dataSendPoint which references VariableDataPrototype_TC12		
Summary	If a dataElement has been received invalidated in case of intra-partition communication and the attribute handleInvalid is set to replace – RTE shall perform the "invalid value substitution" with the initValue. Then the reception will be handled as if a valid value would have been received.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Read API to explicit read the dataElement VariableDataPrototype_TC12.	[SWC] Rte_Read returns RTE_E_OK and provides the initValue.
Step 2	[SWC] Invoke Rte_Invalidate API to invalidate the dataElement VariableDataPrototype_TC12.	[SWC] Rte_Invalidate returns RTE_E_OK indicating invalidation successful.
Step 3	[SWC] Invoke Rte_Read API to explicit read the dataElement VariableDataPrototype_TC12.	[SWC] Rte_Read returns RTE_E_OK and provides initValue.
Post-conditions	NONE	

6.3.25 [ATS_RTE_00645] Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement Of Primitive Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement Of Primitive Data Type		
ID	ATS_RTE_00645	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01231 RTE: SWS_Rte_01282 RTE: SWS_Rte_04505		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC13 typed with a primitive data type with different values as initValue (Ex: 0x12) and invalidValue (Ex: 0xAA) with swImplPolicy ‘standard’ - is typed by a SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC13 A communication matrix with * Signal_TC13 - mapped to VariableDataPrototype_TC13 - transmitted by the ECU		
Summary	In case of inter-ECU communication if the application SW-C has to invalidate a data element of primitive type then it calls the Rte_Invalidate API which in turn will trigger the emission of a signal with the invalidValue.		

Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Invalidate API to invalidate the dataElement VariableDataPrototype_TC13.	[SWC] Rte_Invalidate returns RTE_E_OK.
Step 2	-	[LT] Signal “Signal_TC13” with configured invalidValue has been observed on the bus
Post-conditions	NONE	

6.3.26 [ATS_RTE_00646] Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement With Array Data Type		
ID	ATS_RTE_00646	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01282 RTE: SWS_Rte_05024 RTE: SWS_Rte_05063 RTE: SWS_Rte_05081		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC14array, typed with an array data type, with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA), with swImplPolicy ‘standard’, - is typed by a SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC14array A communication matrix with * SignalGroup_TC14array - mapped to VariableDataPrototype_TC14array - transmitted by the ECU		

Summary	In case of inter-ECU communication if the application SW-C has to invalidate a data element with array type then it calls Rte_Invalidate API which in turn will trigger the emission of a signal with the invalidValue.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Invalidate API to invalidate the array dataElement VariableDataPrototype_TC14array.	[SWC] Rte_Invalidate returns RTE_E_OK indicating invalidation of data.
Step 2	-	[LT] Signal SignalGroup_TC14array with invalidValue has been observed on the bus
Post-conditions	None	

6.3.27 [ATS_RTE_00647] Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement With Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Sender Side For DataElement With Record Data Type		
ID	ATS_RTE_00647	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01206 RTE: SWS_Rte_01207 RTE: SWS_Rte_01282 RTE: SWS_Rte_05024 RTE: SWS_Rte_05063 RTE: SWS_Rte_05081		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC14record, typed with a record data type, with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA), with swImplPolicy ‘standard’, - is typed by a SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC14record A communication matrix with		

	* SignalGroup_TC14record - mapped to VariableDataPrototype_TC14record - transmitted by the ECU
Summary	In case of inter-ECU communication if the application SW-C has to invalidate a data element with record type then it calls Rte_Invalidate API which in turn will trigger the emission of a signal with the invalidValue.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	[SWC] Invoke Rte_Invalidate API to invalidate the record dataElement VariableDataPrototype_TC14record.
Step 2	[SWC] Rte_Invalidate returns RTE_E_OK indicating invalidation of data.
Step 2	-
Post-conditions	None

6.3.28 [ATS_RTE_00648] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataElement Of Primitive Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataElement Of Primitive Data Type		
ID	ATS_RTE_00648	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_01359 RTE: SWS_Rte_02626 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_05026 RTE: SWS_Rte_06011		
Requirements / Reference			

to Test Environment		
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC15 typed with a primitive data type with an invalidValue (Ex: 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC15 (handleInvalid = keep) * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC15 - with a dataReceivePointByArgument which references VariableDataPrototype_TC15 * RunnableEntity_DRError - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC15 - with a dataReceivePointByArgument which references VariableDataPrototype_TC15 A communication matrix with * Signal_TC15 - mapped to VariableDataPrototype_TC15 - received by the ECU	
Summary	If a dataElement has been received invalidated in case of inter-ECU communication and the attribute handleInvalid is set to keep for all receiving software components – the query of the value shall return the received value together with an indication of the invalid status i.e. the return value of Rte_Read shall be RTE_E_INVALID. If a data element has been received invalidated in case of inter-ECU communication and if a DataReceiveErrorEvent is configured for the associated dataElement then the RTE shall activate the runnable "RunnableEntity_DRError" when a signal with invalid value is received.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with Signal_TC15.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read inside to read dataElement VariableDataPrototype_TC15.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the value as sent by LT in step 1.
Step 3	[LT] Transmit a frame with invalid data sent on Signal_TC15.	[RUN<RunnableEntity_DRError>] RunnableEntity_DRError is invoked.
Step 4	[RUN<RunnableEntity_DRError>] Invoke Rte_Read to read dataElement VariableDataPrototype_TC15.	[RUN<RunnableEntity_DRError>] Rte_Read returns RTE_E_INVALID and provides the value as sent by LT in step 3.
Post-conditions	NONE	

6.3.29 [ATS_RTE_00649] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Replace' For DataElement Of Primitive Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Replace' For DataElement Of Primitive Data Type		
ID	ATS_RTE_00649	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_05048 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC16 typed with a primitive data type with different values as initValue (Ex: 0x05) and invalidValue (Ex: 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC16 (handleInvalid = replace) * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC16 - with a dataReceivePointByArgument which references VariableDataPrototype_TC16 A communication matrix with * Signal_TC16 - mapped to VariableDataPrototype_TC16 - received by the ECU		
Summary	If a data element has been received invalidated in case of Inter-ECU communication and the attribute handle Invalid is set to replace for all receiving software components – COM shall be configured to perform the "invalid value substitution" with the initValue. Then the reception will be handled as if a valid value would have been received. i.e. activation of Runnable Entity using the data received event.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps			**Pass Criteria**

Step 1	[LT] Transmit a frame with a valid value onSignal_TC16.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC16.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the value as sent by LT in step 1.
Step 3	[LT] Transmit a frame with invalidValue onSignal_TC16.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 4	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC16.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the initValue.
Post-conditions	NONE	

6.3.30 [ATS_RTE_00650] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataeElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataeElement With Array Data Type		
ID	ATS_RTE_00650	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01289 RTE: SWS_Rte_01359 RTE: SWS_Rte_02626 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_05026 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011 RTE: SWS_Rte_07396 RTE: SWS_Rte_08405		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC17array typed with an array data type with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC17array (handleInvalid = keep) * RunnableEntity_DRError		

	- started by a DataReceiveErrorEvent which references VariableDataPrototype_TC17array - with a dataReceivePointByArgument which references VariableDataPrototype_TC17array A communication matrix with * SignalGroup_TC17array - mapped to VariableDataPrototype_TC17array - received by the ECU
Summary	This test case verifies the explicit inter-ECU data invalidation for a dataElement with array data type on receiver side when the attribute handleInvalid is set as keep.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with invalidValue onSignalGroup_TC17array.	[RUN<RunnableEntity_DRError>] RunnableEntity_DRError is invoked.
Step 2	[RUN<RunnableEntity_DRError>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC17array.	[RUN<RunnableEntity_DRError>] Rte_Read returns RTE_E_INVALID and provides the invalidValue.
Post-conditions	NONE	

6.3.31 [ATS_RTE_00651] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataElement With Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side When Attribute 'HandleInvalid = Keep' For DataElement With Record Data Type		
ID	ATS_RTE_00651	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01289 RTE: SWS_Rte_01359 RTE: SWS_Rte_02626 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_05026 RTE: SWS_Rte_05081 RTE: SWS_Rte_06011 RTE: SWS_Rte_07396 RTE: SWS_Rte_08405		

Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC17record typed with a record data type with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC17record (handleInvalid = keep) * RunnableEntity_DRError - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC17record - with a dataReceivePointByArgument which references VariableDataPrototype_TC17record A communication matrix with * SignalGroup_TC17record - mapped to VariableDataPrototype_TC17record - received by the ECU
Summary	This test case verifies the explicit inter-ECU data invalidation for a dataElement with record data type on receiver side when the attribute handleInvalid is set as keep.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with invalidValue onSignalGroup_TC17record.	[RUN<RunnableEntity_DRError>] RunnableEntity_DRError is invoked.
Step 2	[RUN<RunnableEntity_DRError>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC17record.	[RUN<RunnableEntity_DRError>] Rte_Read returns RTE_E_INVALID and provides the invalidValue.
Post-conditions	NONE	

6.3.32 [ATS_RTE_00652] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side For Attribute 'HandleInvalid = Replace' For DataElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side For Attribute 'HandleInvalid = Replace' For DataElement With Array Data Type		
ID	ATS_RTE_00652	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			

Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_04505 RTE: SWS_Rte_04527 RTE: SWS_Rte_05048	
Requirements / Reference to Test Environment		
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC18array typed with an array data type with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC18array (handleInvalid = replace) * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC18array - with a dataReceivePointByArgument which references VariableDataPrototype_TC18array A communication matrix with * SignalGroup_TC18array - mapped to VariableDataPrototype_TC18array - received by the ECU	
Summary	If a dataElement with array type has been received invalidated in case of inter-ECU communication and the attribute handleInvalid is set to replace for all receiving software components – COM shall be configured to perform the "invalid value substitution" with the initValue. The reception will be handled as if a valid value would have been received i.e. activation of RunnableEntity using the DataReceivedEvent.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with invalid value on SignalGroup_TC18array.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC18array.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the initValue.
Post-conditions	NONE	

6.3.33 [ATS_RTE_00653] Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side For Attribute 'HandleInvalid = Replace' For DataElement With Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Data Invalidation On Receiver Side For Attribute 'HandleInvalid = Replace' For DataElement With Record Data Type		
ID	ATS_RTE_00653	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_04505 RTE: SWS_Rte_04527 RTE: SWS_Rte_05048		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC18record typed with a record data type with different values as initValue (Ex: all subElements = 0x05) and invalidValue (Ex: all subElements = 0xAA) with swImplPolicy 'standard' - SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC18record (handleInvalid = replace) * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC18record - with a dataReceivePointByArgument which references VariableDataPrototype_TC18record A communication matrix with * SignalGroup_TC18record - mapped to VariableDataPrototype_TC18record - received by the ECU		
Summary	If a dataElement with record type has been received invalidated in case of inter-ECU communication and the attribute handleInvalid is set to replace for all receiving software components – COM shall be configured to perform the "invalid value substitution" with the initValue. The reception will be handled as if a valid value would have been received i.e. activation of RunnableEntity using the DataReceivedEvent.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps			**Pass Criteria**

Step 1	[LT] Transmit a frame with invalid value on SignalGroup_TC18record.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC18record.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the initValue.
Post-conditions	NONE	

6.3.34 [ATS_RTE_00654] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement Of Primitive Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement Of Primitive Data Type (Without WaitPoint)		
ID	ATS_RTE_00654	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01084 RTE: SWS_Rte_01086 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_06023 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC19 typed by a primitive data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC19 to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC19 * RunnableEntity_TxAck - with a dataSendPoint which references VariableDataPrototype_TC19 - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint		

	A communication matrix with * Signal_TC19 - mapped to VariableDataPrototype_TC19 * ISignalIPdu_TC19 - Signal_TC19 mapped to ISignalIPdu_TC19 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	To verify explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and activation of a runnable when a TransmissionAcknowledgementRequest is configured for a dataElement with a primitive data type.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC19>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC19.	[RUN<RunnableEntity_TC19>] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Step 2	[RUN<RunnableEntity_TC19>] Invoke Rte_Write API to write the dataElement VariableDataPrototype_TC19.	[RUN<RunnableEntity_TC19>] Rte_Write returns RTE_E_OK.
Step 3	-	[RUN<RunnableEntity_TxAck>] RunnableEntity_TxAck is invoked.
Step 4	[RUN<RunnableEntity_TxAck>] Invoke Rte_Feedback API for the dataElement VariableDataPrototype_TC19.	[RUN<RunnableEntity_TxAck>] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.35 [ATS_RTE_00655] Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest For DataElement Of Primitive Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest For DataElement Of Primitive Data Type (With WaitPoint)		
ID	ATS_RTE_00655	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_01286		

	RTE: SWS_Rte_03532 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_06023 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC20 typed by a primitive data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC20 - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint A communication matrix with * Signal_TC20 - mapped to VariableDataPrototype_TC20 * ISignalIPdu_TC20 - Signal_TC20 mapped to ISignalIPdu_TC20 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies the explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and the wake up of category 2 runnables from a WaitPoint as a result of successful transmission for a primitive data type.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC20.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC20.	[SWC] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.36 [ATS_RTE_00656] Explicit Nonqueued Inter-ECU Transmission error notification For DataElement Of Primitive Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU Transmission error notification For DataElement Of Primitive Data Type (Without WaitPoint)		
ID	ATS_RTE_00656	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2

Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_06023 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC21 typed by a primitive data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC21 to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC21 * RunnableEntity_TxErr - with a dataSendPoint which references VariableDataPrototype_TC21 - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * Signal_TC21 - mapped to VariableDataPrototype_TC21 * ISignalIPdu_TC21 - Signal_TC21 mapped to ISignalIPdu_TC21 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC21 which references ISignalIPdu_TC21		
Summary	This test case verifies that in case of explicit Sender-Receiver communication when an application SW-C calls 'Rte_Feedback' API RTE should provide the acknowledgement status of the transmission to the caller (SW-C). In case of IpduGroup containing Signal_TC21Tx is in Stop mode the transmission will be unsuccessful. In such case of unsuccessful transmission due to stopping of COM IpduGroup Rte_Feedback API returns RTE_E_COM_STOPPED		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ISignalIPduGroup_TC21 shall be in stopped mode.		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	[RUN<RunnableEntity_TC21>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC21.	[RUN<RunnableEntity_TC21>] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	-	[RUN<RunnableEntity_TxErr>] RunnableEntity_TxErr is invoked.
Step 3	[RUN<RunnableEntity_TxErr>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC21.	[RUN<RunnableEntity_TxErr>] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.37 [ATS_RTE_00657] Explicit Nonqueued Inter-ECU Transmission error notification For DataElement Of Primitive Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU Transmission error notification For DataElement Of Primitive Data Type (With WaitPoint)		
ID	ATS_RTE_00657	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC22 typed by a primitive data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC22 - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint (WaitPoint’s timeout equal to the above TransmissionAcknowledgementRequest’s timeout) A communication matrix with * Signal_TC22 - mapped to VariableDataPrototype_TC22 * ISignalIPdu_TC22		

	- Signal_TC22 mapped to ISignalIPdu_TC22 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC22 which references ISignalIPdu_TC22
Summary	This test case is executed when the relevant I-PDU group is stopped. In such case Rte_Write and the following Rte_Feedback return RTE_E_COM_STOPPED without blocking.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ISignalIPduGroup_TC21 shall be in stopped mode.
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC22.
	[SWC] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC22.
	[SWC] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE

6.3.38 [ATS_RTE_00658] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest timeout notification (Without WaitPoint, Primitive Data Type)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest timeout notification (Without WaitPoint, Primitive Data Type)		
ID	ATS_RTE_00658	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_01286 RTE: SWS_Rte_03754 RTE: SWS_Rte_03757 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_06023 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		

Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC23 typed by a primitive data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * RunnableEntity_TC23 to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC23 * RunnableEntity_TxTOut - with a dataSendPoint which references VariableDataPrototype_TC23 - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * Signal_TC23 - mapped to VariableDataPrototype_TC23 * ISignalIPdu_TC23 - Signal_TC23 mapped to ISignalIPdu_TC23 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies that the RTE should support activation of a runnable when a COM callback for TransmissionAcknowledgementRequest timeout notification is invoked by COM and Rte_Feedback API should return RTE_E_TIMEOUT when TransmissionAcknowledgementRequest timeout notification is received from COM.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC23>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC23.	[RUN<RunnableEntity_TC23>] Rte_Write returns RTE_E_OK.
Step 2	-	[RUN<RunnableEntity_TxTOut>] RunnableEntity_TxTOut is invoked.
Step 3	[RUN<RunnableEntity_TxTOut>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC23.	[RUN<RunnableEntity_TxTOut>] Rte_Feedback returns RTE_E_TIMEOUT.
Post-conditions	NONE	

6.3.39 [ATS_RTE_00659] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Primitive Data Type)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Primitive Data Type)

ID	ATS_RTE_00659	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_03532 RTE: SWS_Rte_04505 RTE: SWS_Rte_06023 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC24 typed by a primitive data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC24 - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint (WaitPoint’s timeout equal to the above TransmissionAcknowledgementRequest’s timeout) A communication matrix with * Signal_TC24 - mapped to VariableDataPrototype_TC24 * ISignalIPdu_TC24 - Signal_TC24 mapped to ISignalIPdu_TC24 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU		
Summary	This test case verifies that if a callback for TransmissionAcknowledgementRequest timeout notification is configured then the RTE should support to wake up category 2 runnables from a WaitPoint as a result of COM callback for TransmissionAcknowledgementRequest timeout (Rte_Feedback does not stay blocked).		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC24.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC24.	[SWC] Rte_Feedback returns RTE_E_TIMEOUT.
Post-conditions	NONE	

6.3.40 [ATS_RTE_00660] Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement Of Primitive Data Type

Test Objective	Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement Of Primitive Data Type		
ID	ATS_RTE_00660	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_01359 RTE: SWS_Rte_02703 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_05022		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype to receive - VariableDataPrototype_TC25 typed with a primitive data type with swImplPolicy 'standard' - NonqueuedReceiverComSpec with aliveTimeout > 0 and handleTimeout = none * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC25 - with a dataReceivePointByArgument which references VariableDataPrototype_TC25 * RunnableEntity_AliveTOut - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC25 - with a dataReceivePointByArgument which references VariableDataPrototype_TC25 A communication matrix with * Signal_TC25 - mapped to VariableDataPrototype_TC25 - received by the ECU		

Summary	This test case verifies that the RTE supports the activation of a runnable when a signal reception timeout occurs.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame periodically with Signal_TC25.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC25.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the value sent periodically.
Step 3	[LT] Stop sending frame containing Signal_TC25 for more than the configured aliveTimeout.	[RUN<RunnableEntity_AliveTOut>] RunnableEntity_AliveTOut is invoked.
Step 4	[RUN<RunnableEntity_AliveTOut>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC25.	[RUN<RunnableEntity_AliveTOut>] Rte_Read returns RTE_E_MAX_AGE_EXCEEDED and provides the last value sent periodically (step 1).
Post-conditions	NONE	

6.3.41 [ATS_RTE_00661] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Array Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Array Data Type (Without WaitPoint)		
ID	ATS_RTE_00661	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01086 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527		

	RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC26array typed by an array data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC26array to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC26array * RunnableEntity_TxAck - with a dataSendPoint which references VariableDataPrototype_TC26array - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC26array - mapped to VariableDataPrototype_TC26array * ISignalIPdu_TC26array - SignalGroup_TC26array mapped to ISignalIPdu_TC26array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies the explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and activation of a runnable on a DataSendCompletedEvent for a dataElement with an array type.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC26array>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC26array.	[RUN<RunnableEntity_TC26array>] Rte_Write returns RTE_E_OK.
Step 2	-	[RUN<RunnableEntity_TxAck>] RunnableEntity_TxAck is invoked.
Step 3	[RUN<RunnableEntity_TxAck>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC26array.	[RUN<RunnableEntity_TxAck>] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.42 [ATS_RTE_00662] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Record Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Record Data Type (Without WaitPoint)		
ID	ATS_RTE_00662	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01086 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC26record typed by a record data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC26record to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC26record * RunnableEntity_TxAck - with a dataSendPoint which references VariableDataPrototype_TC26record - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC26record - mapped to VariableDataPrototype_TC26record * ISignalIPdu_TC26record - SignalGroup_TC26record mapped to ISignalIPdu_TC26record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU		
Summary	This test case verifies the explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and activation of a runnable on a DataSendCompletedEvent for a dataElement with a record type.		

Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC26record>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC26record.	[RUN<RunnableEntity_TC26record>] Rte_Write returns RTE_E_OK.
Step 2	-	[RUN<RunnableEntity_TxAck>] RunnableEntity_TxAck is invoked.
Step 3	[RUN<RunnableEntity_TxAck>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC26record.	[RUN<RunnableEntity_TxAck>] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.43 [ATS_RTE_00663] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement WIth Array Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement WIth Array Data Type (With WaitPoint)		
ID	ATS_RTE_00663	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01086 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_03532 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which		

	- sends VariableDataPrototype_TC27array typed by an array data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC27array - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint A communication matrix with * SignalGroup_TC27array - mapped to VariableDataPrototype_TC27array * ISignalIPdu_TC27array - SignalGroup_TC27array mapped to ISignalIPdu_TC27array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies the explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and to wake up category 2 runnables from a WaitPoint as a result of successful transmission of an array.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC27array. [SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC27array. [SWC] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE

6.3.44 [ATS_RTE_00664] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Record Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest For DataElement With Record Data Type (With WaitPoint)		
ID	ATS_RTE_00664	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01086		

	RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_03532 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07820 RTE: SWS_Rte_07824
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC27record typed by a record data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC27record - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint A communication matrix with * SignalGroup_TC27record - mapped to VariableDataPrototype_TC27record * ISignalIPdu_TC27record - SignalGroup_TC27record mapped to ISignalIPdu_TC27record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies the explicit nonqueued inter-ECU TransmissionAcknowledgementRequest and to wake up category 2 runnables from a WaitPoint as a result of successful transmission of a record.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC27record.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC27record.	[SWC] Rte_Feedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.45 [ATS_RTE_00665] Explicit Nonqueued Inter-ECU Transmission Error Notification For DataElement With Array Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU Transmission Error Notification For DataElement With Array Data Type (Without WaitPoint)		
ID	ATS_RTE_00665	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC28array typed by an array data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC28array to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC28array * RunnableEntity_TxErr - with a dataSendPoint which references VariableDataPrototype_TC28array - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC28array - mapped to VariableDataPrototype_TC28array * ISignalIPdu_TC28array - SignalGroup_TC28array mapped to ISignalIPdu_TC28array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC28array which references ISignalIPdu_TC28array		
Summary	This test case verifies that in case of explicit Sender-Receiver communication when an application SW-C calls 'Rte_Feedback' API RTE should provide the acknowledgement status of the transmission to the caller (SW-C). This test case forces a failure of the transmission by stopping the relevant I-PDU group.		

Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC28array>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC28array.	[RUN<RunnableEntity_TC28array>] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	-	[RUN<RunnableEntity_TxErr>] RunnableEntity_TxErr is invoked.
Step 3	[RUN<RunnableEntity_TxErr>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC28array.	[RUN<RunnableEntity_TxErr>] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.46 [ATS_RTE_00666] Explicit Nonqueued Inter-ECU Transmission Error Notification For DataElement With Record Data Type (Without WaitPoint)

Test Objective	Explicit Nonqueued Inter-ECU Transmission Error Notification For DataElement With Record Data Type (Without WaitPoint)		
ID	ATS_RTE_00666	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC28record typed by a record data type with swImplPolicy ‘standard’		

	- has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC28record to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC28record * RunnableEntity_TxErr - with a dataSendPoint which references VariableDataPrototype_TC28record - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC28record - mapped to VariableDataPrototype_TC28record * ISignalIPdu_TC28record - SignalGroup_TC28record mapped to ISignalIPdu_TC28record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC28record which references ISignalIPdu_TC28record
Summary	This test case verifies that in case of explicit Sender-Receiver communication when an application SW-C calls 'Rte_Feedback' API RTE should provide the acknowledgement status of the transmission to the caller (SW-C). This test case forces a failure of the transmission by stopping the relevant I-PDU group.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC28record>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC28record.	[RUN<RunnableEntity_TC28record>] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	-	[RUN<RunnableEntity_TxErr>] RunnableEntity_TxErr is invoked.
Step 3	[RUN<RunnableEntity_TxErr>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC28record.	[RUN<RunnableEntity_TxErr>] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.47 [ATS_RTE_00667] Explicit Nonqueued inter-ECU Transmission error notification For DataElement With Array Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued inter-ECU Transmission error notification For DataElement With Array Data Type (With WaitPoint)		
ID	ATS_RTE_00667	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			

Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC29array typed by an array data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC29array - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint A communication matrix with * SignalGroup_TC29array - mapped to VariableDataPrototype_TC29array * ISignalIPdu_TC29array - SignalGroup_TC29array mapped to ISignalIPdu_TC29array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC29array which references ISignalIPdu_TC29array
Summary	This test case is executed when the relevant I-PDU group is stopped. In such case Rte_Write and the following Rte_Feedback return RTE_E_COM_STOPPED without blocking.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION state ISignalIPduGroup_TC29array shall be in stopped mode.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write API to write the dataElement VariableDataPrototype_TC29array.	[SWC] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC29array.	[RTE] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.48 [ATS_RTE_00668] Explicit Nonqueued inter-ECU Transmission error notification For DataElement With Record Data Type (With WaitPoint)

Test Objective	Explicit Nonqueued inter-ECU Transmission error notification For DataElement With Record Data Type (With WaitPoint)		
ID	ATS_RTE_00668	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_04526 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636 RTE: SWS_Rte_07822 RTE: SWS_Rte_07824		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC29record typed by a record data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC29record - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint A communication matrix with * SignalGroup_TC29record - mapped to VariableDataPrototype_TC29record * ISignalIPdu_TC29record - SignalGroup_TC29record mapped to ISignalIPdu_TC29record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU * ISignalIPduGroup_TC29record which references ISignalIPdu_TC29record		
Summary	This test case is executed when the relevant I-PDU group is stopped. In such case Rte_Write and the following Rte_Feedback return RTE_E_COM_STOPPED without blocking.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION state ISignalIPduGroup_TC29record shall be in stopped mode.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write API to write the dataElement VariableDataPrototype_TC29record.	[SWC] Rte_Write returns RTE_E_COM_STOPPED.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC29record.	[RTE] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.49 [ATS_RTE_00669] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest Timeout Notification (Without WaitPoint, Array Data Type)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest Timeout Notification (Without WaitPoint, Array Data Type)		
ID	ATS_RTE_00669	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_06011 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC30array typed by an array data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * RunnableEntity_TC30array to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC30array		

	* RunnableEntity_TxTOut - with a dataSendPoint which references VariableDataPrototype_TC30array - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC30array - mapped to VariableDataPrototype_TC30array * ISignalIPdu_TC30array - SignalGroup_TC30array mapped to ISignalIPdu_TC30array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies that the RTE should support activation of a runnable when a COM callback for TransmissionAcknowledgementRequest timeout notification is invoked by COM and Rte_Feedback API should return RTE_E_TIMEOUT when TransmissionAcknowledgementRequest timeout notification is received from COM.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC30array>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC30array.	[RUN<RunnableEntity_TC30array>] Rte_Write returns RTE_E_OK.
Step 2	-	[RUN<RunnableEntity_TxTOut>] RunnableEntity_TxTOut is invoked.
Step 3	[RUN<RunnableEntity_TxTOut>] Invoke Rte_Feedback API for the dataElement VariableDataPrototype_TC30array.	[RUN<RunnableEntity_TxTOut>] Rte_Feedback returns RTE_E_TIMEOUT.
Post-conditions	NONE	

6.3.50 [ATS_RTE_00670] Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest Timeout Notification (Without WaitPoint, Record Data Type)

Test Objective	Explicit Nonqueued Inter-ECU TransmissionAcknowledgementRequest Timeout Notification (Without WaitPoint, Record Data Type)		
ID	ATS_RTE_00670	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080		

	RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01285 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_06011 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC30record typed by a record data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * RunnableEntity_TC30record to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC30record * RunnableEntity_TxTOut - with a dataSendPoint which references VariableDataPrototype_TC30record - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * SignalGroup_TC30record - mapped to VariableDataPrototype_TC30record * ISignalIPdu_TC30record - SignalGroup_TC30record mapped to ISignalIPdu_TC30record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies that the RTE should support activation of a runnable when a COM callback for TransmissionAcknowledgementRequest timeout notification is invoked by COM and Rte_Feedback API should return RTE_E_TIMEOUT when TransmissionAcknowledgementRequest timeout notification is received from COM.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC30record>] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC30record.	[RUN<RunnableEntity_TC30record>] Rte_Write returns RTE_E_OK.
Step 2	-	[RUN<RunnableEntity_TxTOut>] RunnableEntity_TxTOut is invoked.
Step 3	[RUN<RunnableEntity_TxTOut>] Invoke Rte_Feedback API for the dataElement VariableDataPrototype_TC30record.	[RUN<RunnableEntity_TxTOut>] Rte_Feedback returns RTE_E_TIMEOUT.

Post-conditions	NONE

6.3.51 [ATS_RTE_00671] Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Array Data Type)

Test Objective	Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Array Data Type)		
ID	ATS_RTE_00671	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_03532 RTE: SWS_Rte_06011 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC31array typed by an array data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC31array - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint (WaitPoint’s timeout equal to the above TransmissionAcknowledgementRequest’s timeout) A communication matrix with * SignalGroup_TC31array - mapped to VariableDataPrototype_TC31array * ISignalIPdu_TC31array - SignalGroup_TC31array mapped to ISignalIPdu_TC31array with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU		
Summary	This test case verifies that if a callback for TransmissionAcknowledgementRequest timeout notification is configured then the RTE should support to wake up category 2 runnables from a WaitPoint as a result of COM callback for		

	TransmissionAcknowledgementRequest timeout (Rte_Feedback does not stay blocked).	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC31array.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC31array.	[SWC] Rte_Feedback returns RTE_E_TIMEOUT.
Post-conditions	NONE	

6.3.52 [ATS_RTE_00672] Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Record Data Type)

Test Objective	Explicit Nonqueued inter-ECU TransmissionAcknowledgementRequest timeout notification (With WaitPoint, Record Data Type)		
ID	ATS_RTE_00672	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01280 RTE: SWS_Rte_01283 RTE: SWS_Rte_01284 RTE: SWS_Rte_03532 RTE: SWS_Rte_06011 RTE: SWS_Rte_07637 RTE: SWS_Rte_07820		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC31record typed by a record data type with swImplPolicy ‘standard’ - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0		

	* a RunnableEntity to execute the test sequence - with a dataSendPoint which references VariableDataPrototype_TC31record - with a WaitPoint that references a DataSendCompletedEvent which references the previous dataSendPoint (WaitPoint's timeout equal to the above TransmissionAcknowledgementRequest's timeout) A communication matrix with * SignalGroup_TC31record - mapped to VariableDataPrototype_TC31record * ISignalIPdu_TC31record - SignalGroup_TC31record mapped to ISignalIPdu_TC31record with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	This test case verifies that if a callback for TransmissionAcknowledgementRequest timeout notification is configured then the RTE should support to wake up category 2 runnables from a WaitPoint as a result of COM callback for TransmissionAcknowledgementRequest timeout (Rte_Feedback does not stay blocked).
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write the dataElement VariableDataPrototype_TC31record.
	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC31record.
	[SWC] Rte_Feedback returns RTE_E_TIMEOUT.
Post-conditions	NONE

6.3.53 [ATS_RTE_00673] Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement With Array Data Type

Test Objective	Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement With Array Data Type		
ID	ATS_RTE_00673	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292		

	RTE: SWS_Rte_01359 RTE: SWS_Rte_02703 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_05022
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * an RPortPrototype to receive - VariableDataPrototype_TC32array typed with an array data type with swImplPolicy 'standard' with an initValue (e.g. all subElements = 0x05) - NonqueuedReceiverComSpec with aliveTimeout > 0 and handleTimeoutType = replace * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC32array - with a dataReceivePointByArgument which references VariableDataPrototype_TC32array * RunnableEntity_AliveTOut - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC32array - with a dataReceivePointByArgument which references VariableDataPrototype_TC32array A communication matrix with * SignalGroup_TC32array - mapped to VariableDataPrototype_TC32array - received by the ECU
Summary	This test case verifies that the RTE supports the activation of a runnable when a signal group reception timeout occurs.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with SignalGroup_TC32array.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC32array.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the value sent periodically.
Step 3	[LT] Stop sending frame containing SignalGroup_TC32array for more than configured aliveTimeout.	[RUN<RunnableEntity_AliveTOut>] RunnableEntity_AliveTOut is invoked.
Step 4	[RUN<RunnableEntity_AliveTOut>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC32array.	[RUN<RunnableEntity_AliveTOut>] Rte_Read returns RTE_E_MAX_AGE_EXCEEDED and provides the initValue.
Post-conditions	NONE	

6.3.54 [ATS_RTE_00674] Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement With Record Data Type

Test Objective	Explicit Nonqueued Inter-ECU Reception On AliveTimeout Expiry For DataElement With Record Data Type		
ID	ATS_RTE_00674	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_01359 RTE: SWS_Rte_02703 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_05022		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype to receive - VariableDataPrototype_TC32record typed with a record data type with swImplPolicy ‘standard’ with an initValue (e.g. all subElements = 0x05) - NonqueuedReceiverComSpec with aliveTimeout > 0 and handleTimeout = replace * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC32record - with a dataReceivePointByArgument which references VariableDataPrototype_TC32record * RunnableEntity_AliveTOut - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC32record - with a dataReceivePointByArgument which references VariableDataPrototype_TC32record A communication matrix with * SignalGroup_TC32record - mapped to VariableDataPrototype_TC32record - received by the ECU		
Summary	This test case verifies that the RTE supports the activation of a runnable when a signal group reception timeout occurs.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with SignalGroup_TC32record.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC32record.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the value sent periodically.
Step 3	[LT] Stop sending frame containing SignalGroup_TC32record for more than configured aliveTimeout.	[RUN<RunnableEntity_AliveTOut>] RunnableEntity_AliveTOut is invoked.
Step 4	[RUN<RunnableEntity_AliveTOut>] Invoke Rte_Read to read the dataElement VariableDataPrototype_TC32record.	[RUN<RunnableEntity_AliveTOut>] Rte_Read returns RTE_E_MAX_AGE_EXCEEDED and provides the initValue.
Post-conditions	NONE	

6.3.55 [ATS_RTE_00675] Explicit Nonqueued Intra-Partition Rte_DRead

Test Objective	Explicit Nonqueued Intra-Partition Rte_DRead		
ID	ATS_RTE_00675	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01280 RTE: SWS_Rte_07394 RTE: SWS_Rte_07395 RTE: SWS_Rte_07820		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype to send - VariableDataPrototype_TC33 typed by a primitive data type with swImplPolicy ‘standard’ * An RPortPrototype to receive VariableDataPrototype_TC33 - connected with the previous PPortPrototype * A RunnableEntity to execute the test sequence - with a dataReceivePointByValue which references VariableDataPrototype_TC33 - with a dataSendPoint which references VariableDataPrototype_TC33		
Summary	A dataElement is written and later read using Rte_DRead. The data read shall be same as the data written.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_DRead.	[SWC] Rte_DRead returns the data which was written in step 1.
Post-conditions	NONE	

6.3.56 [ATS_RTE_00676] Explicit Nonqueued Inter-ECU Rte_DRead

Test Objective	Explicit Nonqueued Inter-ECU Rte_DRead		
ID	ATS_RTE_00676	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01135 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011 RTE: SWS_Rte_07394 RTE: SWS_Rte_07395		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A RPortPrototype to receive - VariableDataPrototype_TC34 typed by a primitive data type with swImplPolicy ‘standard’ * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC34 - with a dataReceivePointByValue which references VariableDataPrototype_TC34 A communication matrix with * Signal_TC34 - mapped to VariableDataPrototype_TC34 - received by the ECU		
Summary	A signal is sent on the bus which triggers a runnable. The runnable read the VariableDataPrototype associated to the signal with an Rte_DRead API. The test checks that the data read is the same as sent on the bus.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit a frame with Signal_TC34.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_DRead.	[RUN<RunnableEntity_DRE>] Rte_DRead returns the data which was sent in step 1.
Post-conditions	NONE	

6.3.57 [ATS_RTE_00677] Queued Intra-partition Rte_Send And Rte_Receive

Test Objective	Queued Intra-partition Rte_Send And Rte_Receive		
ID	ATS_RTE_00677	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_01092 RTE: SWS_Rte_01094 RTE: SWS_Rte_01281 RTE: SWS_Rte_01288 RTE: SWS_Rte_02633 RTE: SWS_Rte_06011 RTE: SWS_Rte_07673		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype to send - VariableDataPrototype_TC35 typed by a primitive data type with swImplPolicy ‘queued’ * An RPortPrototype to receive VariableDataPrototype_TC35 - connected with the previous PPortPrototype * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC35 - with a dataSendPoint which references VariableDataPrototype_TC35		
Summary	The test checks the behaviour of Rte_Receive before any event was sent and checks that an event can be sent and received correctly.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Receive for the dataElement VariableDataPrototype_TC35.	[RTE] Rte_Receive returns RTE_E_NO_DATA and does not change the buffer provided as argument.
Step 2	[SWC] Invoke Rte_Send for the dataElement VariableDataPrototype_TC35.	[SWC] Rte_Send returns RTE_E_OK.
Step 3	[SWC] Invoke Rte_Receive for the dataElement VariableDataPrototype_TC35.	[SWC] Rte_Receive returns RTE_E_OK and provides the data (event) sent in step 2.
Post-conditions	NONE	

6.3.58 [ATS_RTE_00678] Queued Inter-ECU Transmission With Successful Acknowledgement Request

Test Objective	Queued Inter-ECU Transmission With Successful Acknowledgement Request		
ID	ATS_RTE_00678	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_01083 RTE: SWS_Rte_01086 RTE: SWS_Rte_01137 RTE: SWS_Rte_01281 RTE: SWS_Rte_01283 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC36 typed by a primitive data type with swImplPolicy ‘queued’ - has a QueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC36 to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC36 * RunnableEntity_TxAck - with a dataSendPoint which references VariableDataPrototype_TC36 - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint		

	A communication matrix with * Signal_TC36 - mapped to VariableDataPrototype_TC36 * ISignalIPdu_TC36 - Signal_TC36 mapped to ISignalIPdu_TC36 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	The test checks that the acknowledgment of an event (queued) inter-ECU transmission can activate a runnable and that the Rte_Feedback API provides the acknowledgement status.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC36>] Invoke Rte_Send for the dataElement VariableDataPrototype_TC36.
Step 2	-
Step 3	-
Step 4	[RUN<RunnableEntity_TxAck>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC36.
Post-conditions	NONE

6.3.59 [ATS_RTE_00679] Queued Inter-ECU Transmission Error Notification

Test Objective	Queued Inter-ECU Transmission Error Notification		
ID	ATS_RTE_00679	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01072 RTE: SWS_Rte_01080 RTE: SWS_Rte_01083 RTE: SWS_Rte_01137 RTE: SWS_Rte_01281 RTE: SWS_Rte_01283 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_06011 RTE: SWS_Rte_07636		

Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC37 typed by a primitive data type with swImplPolicy 'queued' - has a QueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC37 to initiate the test sequence - with a dataSendPoint which references VariableDataPrototype_TC37 * RunnableEntity_TxErr - with a dataSendPoint which references VariableDataPrototype_TC37 - started by a DataSendCompletedEvent which references this dataSendPoint and is not referenced by a WaitPoint A communication matrix with * Signal_TC37 - mapped to VariableDataPrototype_TC37 * ISignalIPdu_TC37 - Signal_TC37 mapped to ISignalIPdu_TC37 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	The test is executed in COMM_NO_COMMUNICATION mode to trigger a transmission failure. The test checks that transmission error of event can activate a runnable and that the Rte_Feedback API provides the error status RTE_E_COM_STOPPED.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC37>] Invoke Rte_Send for the dataElement VariableDataPrototype_TC37.	[RUN<RunnableEntity_TC37>] Rte_Send returns RTE_E_COM_STOPPED.
Step 2	-	[RUN<RunnableEntity_TxErr>] RunnableEntity_TxAck is invoked.
Step 3	[RUN<RunnableEntity_TxErr>] Invoke Rte_Feedback for the dataElement VariableDataPrototype_TC37.	[RUN<RunnableEntity_TxErr>] Rte_Feedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.60 [ATS_RTE_00680] Queued Inter-ECU Rte_Receive For DataElement Of Primitive Data Type

Test Objective	Queued Inter-ECU Rte_Receive For DataElement Of Primitive Data Type		
ID	ATS_RTE_00680	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_01092 RTE: SWS_Rte_01135 RTE: SWS_Rte_01292 RTE: SWS_Rte_02598 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04505 RTE: SWS_Rte_06011
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC38 typed by a primitive data type with swImplPolicy ‘queued’ * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC38 - with a dataReceivePointByArgument which references VariableDataPrototype_TC38 A communication matrix with * Signal_TC38 - mapped to VariableDataPrototype_TC38 - received by the ECU
Summary	The test checks that in case of successful reception of a dataElement of primitive data type in a queued inter-ECU communication a DataReceivedEvent can activate a RunnableEntity and the Rte_Receive API provides the received event.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Transmit frame withSignal_TC38.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke the Rte_Receive API for the dataElement VariableDataPrototype_TC38.	[RUN<RunnableEntity_DRE>] Rte_Receive returns RTE_E_OK and provides the data sent in Signal_TC38.
Post-conditions	NONE	

6.3.61 [ATS_RTE_00681] Queued Inter-ECU Rte_Receive For DataElement With Array Data Type

Test Objective	Queued Inter-ECU Rte_Receive For DataElement With Array Data Type

ID	ATS_RTE_00681	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01092 RTE: SWS_Rte_01135 RTE: SWS_Rte_01292 RTE: SWS_Rte_02598 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC39array typed by an array data type with swImplPolicy 'queued' * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC39array - with a dataReceivePointByArgument which references VariableDataPrototype_TC39array A communication matrix with * SignalGroup_TC39array - mapped to VariableDataPrototype_TC39array - received by the ECU		
Summary	The test checks that in case of successful reception of a dataElement with array data type in a queued inter-ECU communication a DataReceivedEvent can activate a RunnableEntity and the Rte_Receive API provides the received event.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	[LT] Transmit frame with SignalGroup_TC39array.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.	
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Receive for the dataElement VariableDataPrototype_TC39array.	[RUN<RunnableEntity_DRE>] Rte_Receive returns RTE_E_OK and provides the data sent in SignalGroup_TC39array.	
Post-conditions	NONE		

6.3.62 [ATS_RTE_00682] Queued Inter-ECU Rte_Receive For DataElement With Record Data Type

Test Objective	Queued Inter-ECU Rte_Receive For DataElement With Record Data Type		
ID	ATS_RTE_00682	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01092 RTE: SWS_Rte_01135 RTE: SWS_Rte_01292 RTE: SWS_Rte_02598 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_04527 RTE: SWS_Rte_06011		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC39record typed by a record data type with swImplPolicy 'queued' * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC39record - with a dataReceivePointByArgument which references VariableDataPrototype_TC39record A communication matrix with * SignalGroup_TC39record - mapped to VariableDataPrototype_TC39record - received by the ECU		
Summary	The test checks that in case of successful reception of a dataElement with record data type in a queued inter-ECU communication a DataReceivedEvent can activate a RunnableEntity and the Rte_Receive API provides the received event.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[LT] Transmit frame with SignalGroup_TC39record.		[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 2	[RUN<RunnableEntity_DRE>] Invoke Rte_Receive for the dataElement VariableDataPrototype_TC39record.		[RUN<RunnableEntity_DRE>] Rte_Receive returns RTE_E_OK and provides the data sent in SignalGroup_TC39record.

Post-conditions	NONE

6.3.63 [ATS_RTE_00683] Inter-ECU Rte_IsUpdated API Functionality

Test Objective	Inter-ECU Rte_IsUpdated API Functionality		
ID	ATS_RTE_00683	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01135 RTE: SWS_Rte_01289 RTE: SWS_Rte_01292 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_07385 RTE: SWS_Rte_07386 RTE: SWS_Rte_07390 RTE: SWS_Rte_07391 RTE: SWS_Rte_07392 RTE: SWS_Rte_07393		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which receives - VariableDataPrototype_TC41 typed by a primitive data type with swImplPolicy ‘standard’ - NonqueuedReceiverComSpec with enableUpdate = TRUE * RunnableEntity_TC41 to initiate the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC41 * RunnableEntity_DRE - started by a DataReceivedEvent on reception of VariableDataPrototype_TC41 - with a dataReceivePointByArgument which references VariableDataPrototype_TC41 A communication matrix with * Signal_TC41 - mapped to VariableDataPrototype_TC41 - received by the ECU		
Summary	This test checks that after reception of a signal configured for an inter-ECU reception with the enableUpdate attribute set to TRUE the corresponding enable flag is provided by the Rte_IsUpdated API and that subsequent calls to Rte_IsUpdated without signal reception return FALSE.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC41>] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC41.	[RUN<RunnableEntity_TC41>] Rte_IsUpdated returns FALSE
Step 2	[LT] Transmit a frame with Signal_TC41.	[RUN<RunnableEntity_DRE>] RunnableEntity_DRE is invoked.
Step 3	[RUN<RunnableEntity_DRE>] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC41.	[RUN<RunnableEntity_DRE>] Rte_IsUpdated returns TRUE.
Step 4	[RUN<RunnableEntity_DRE>] Invoke Rte_Read for the dataElement VariableDataPrototype_TC41.	[RUN<RunnableEntity_DRE>] Rte_Read returns RTE_E_OK and provides the Signal_TC41 value sent in step 2.
Step 5	[RUN<RunnableEntity_DRE>] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC41.	[RUN<RunnableEntity_DRE>] Rte_IsUpdated returns FALSE.
Post-conditions	NONE	

6.3.64 [ATS_RTE_00684] Intra-Partition Rte_IsUpdated API Functionality

Test Objective	Intra-Partition Rte_IsUpdated API Functionality		
ID	ATS_RTE_00684	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01071 RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01280 RTE: SWS_Rte_01289 RTE: SWS_Rte_06011 RTE: SWS_Rte_07385 RTE: SWS_Rte_07386 RTE: SWS_Rte_07390 RTE: SWS_Rte_07391 RTE: SWS_Rte_07392 RTE: SWS_Rte_07393 RTE: SWS_Rte_07820		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype to send - VariableDataPrototype_TC42 typed by a primitive data type with swImplPolicy		

	'standard' * An RPortPrototype to receive VariableDataPrototype_TC42 - connected with the previous PPortPrototype - NonqueuedReceiverComSpec with enableUpdate = TRUE * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC42 - with a dataSendPoint which references VariableDataPrototype_TC42
Summary	To verify setting of update flag of the dataElements configured with the "enableUpdate" attribute in case explicit nonqueued intra-partition reception (Rte_Read). This test checks that in case of intra-partition communication with the enableUpdate set to TRUE on the receiver side after transmission of a dataElement the corresponding enable flag is provided by the Rte_IsUpdated API and that subsequent calls to Rte_IsUpdated without transmission of the DataElement return FALSE.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC42.	[SWC] Rte_IsUpdated returns FALSE.
Step 2	[SWC] Invoke Rte_Write for the dataElement VariableDataPrototype_TC42.	[SWC] Rte_Write returns RTE_E_OK.
Step 3	[SWC] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC42.	[SWC] Rte_IsUpdated returns TRUE.
Step 4	[SWC] Invoke Rte_Read for the dataElement VariableDataPrototype_TC42.	[SWC] Rte_Read returns RTE_E_OK and provides the value written in step 1.
Step 5	[SWC] Invoke Rte_IsUpdated for the dataElement VariableDataPrototype_TC42.	[SWC] Rte_IsUpdated returns FALSE.
Post-conditions	NONE	

6.3.65 [ATS_RTE_00685] Implicit Write For Intra-Partition Sender Receiver Communication Using Rte_IWriteRef API

Test Objective	Implicit Write For Intra-Partition Sender Receiver Communication Using Rte_IWriteRef API		
ID	ATS_RTE_00685	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement			

on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_01301 RTE: SWS_Rte_03741 RTE: SWS_Rte_05509 RTE: SWS_Rte_05510 RTE: SWS_Rte_05511 RTE: SWS_Rte_06004 RTE: SWS_Rte_06011
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * A PPortPrototype to send - VariableDataPrototype_TC45 with swImplPolicy ‘standard’ * An RPortPrototype to receive VariableDataPrototype_TC45 - connected with the previous PPortPrototype * RunnableEntity_TC45 to execute the test sequence - with a dataReadAccess which references VariableDataPrototype_TC45 - with a dataWriteAccess which references VariableDataPrototype_TC45
Summary	This test case verifies implicit intra-partition Sender Receiver communication using Rte_IWriteRef API to write the value into the dataElement and Rte_IRead to read the value written into the dataElement.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC45>] Invoke Rte_IWriteRef to write the dataElement VariableDataPrototype_TC45. Write some data (Ex:0x20) in the buffer provided by Rte_IWriteRef.	-
Step 2	[RUN<RunnableEntity_TC45>] Return to exit from this RunnableEntity_TC45 execution-instance.	-
Step 3	[CP] Wait for the next execution-instance of RunnableEntity_TC45.	[RUN<RunnableEntity_TC45>] RunnableEntity_TC45 is invoked
Step 4	[RUN<RunnableEntity_TC45>] Invoke Rte_IRead to read the dataElement VariableDataPrototype_TC45.	[RUN<RunnableEntity_TC45>] Rte_IRead returns the data which was written in step 1 (Ex: 0x20).
Post-conditions	NONE	

6.3.66 [ATS_RTE_00686] Rte_IFeedback API Functionality For Successful Transmission

Test Objective	Rte_IFeedback API Functionality For Successful Transmission

ID	ATS_RTE_00686	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01302 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_03744 RTE: SWS_Rte_03746 RTE: SWS_Rte_04505 RTE: SWS_Rte_07367 RTE: SWS_Rte_07376 RTE: SWS_Rte_07379 RTE: SWS_Rte_07647 RTE: SWS_Rte_07648		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC49 typed by a primitive data type with swImplPolicy ‘standard’ - has a QueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC49 to initiate the test sequence - with a dataWriteAccess which references VariableDataPrototype_TC49 * RunnableEntity_IFeedBackTC49 - with a dataWriteAccess which references VariableDataPrototype_TC49 - started by a DataWriteCompletedEvent which references this dataWriteAccess A communication matrix with * Signal_TC49 - mapped to VariableDataPrototype_TC49 * ISignalIPdu_TC49 - Signal_TC49 mapped to ISignalIPdu_TC49 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU		
Summary	The test checks that after a successful transmission with Rte_IWrite a DataWriteCompletedEvent is triggered and the Rte_IFeedback provides access to the transmission acknowledgement.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	[RUN<RunnableEntity_TC49>] Invoke Rte_IWrite to write the dataElement VariableDataPrototype_TC49.	-
Step 2	[RUN<RunnableEntity_TC49>] Return to exit from this RunnableEntity_TC49 execution-instance.	[LT] A frame with Signal_TC49 is observed on the bus with the value written in step 1.
Step 3	-	[RUN<RunnableEntity_IFeedBackTC49>] RunnableEntity_IFeedBackTC49 is invoked.
Step 4	[RUN<RunnableEntity_IFeedBackTC49>] Invoke Rte_IFeedback for the dataElement VariableDataPrototype_TC49.	[RUN<RunnableEntity_IFeedBackTC49>] Rte_IFeedback returns RTE_E_TRANSMIT_ACK.
Post-conditions	NONE	

6.3.67 [ATS_RTE_00687] Rte_IFeedback API Functionality For Transmission Error

Test Objective	Rte_IFeedback API Functionality For Transmission Error		
ID	ATS_RTE_00687	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01302 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_03744 RTE: SWS_Rte_03746 RTE: SWS_Rte_04505 RTE: SWS_Rte_07367 RTE: SWS_Rte_07375 RTE: SWS_Rte_07379 RTE: SWS_Rte_07647 RTE: SWS_Rte_07648		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC50 typed by a primitive data type with swImplPolicy ‘standard’ - has a QueuedSenderComSpec with a TransmissionAcknowledgementRequest * RunnableEntity_TC50 to initiate the test sequence - with a dataWriteAccess which references VariableDataPrototype_TC50 * RunnableEntity_IFeedBackTC50 - with a dataWriteAccess which references VariableDataPrototype_TC50 - started by a DataWriteCompletedEvent which references this dataWriteAccess		

	A communication matrix with * Signal_TC50 - mapped to VariableDataPrototype_TC50 * ISignalIPdu_TC50 - Signal_TC50 mapped to ISignalIPdu_TC50 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	The test checks that after a failed transmission (due to COMM_NO_COMMUNICATION mode) with Rte_IWrite a DataWriteCompletedEvent is triggered and the Rte_IFeedback provides access to the transmission error.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in COMM_NO_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC50>] Invoke Rte_IWrite to write the dataElement VariableDataPrototype_TC50.	-
Step 2	[RUN<RunnableEntity_TC50>] Return to exit from this RunnableEntity_TC50 execution-instance.	[RUN<RunnableEntity_IFeedBackTC50>] RunnableEntity_IFeedBackTC50 is invoked.
Step 3	[RUN<RunnableEntity_IFeedBackTC50>] Invoke Rte_IFeedback for the dataElement VariableDataPrototype_TC50.	[RUN<RunnableEntity_IFeedBackTC50>] Rte_IFeedback returns RTE_E_COM_STOPPED.
Post-conditions	NONE	

6.3.68 [ATS_RTE_00688] Rte_IFeedback API Functionality For Transmission Acknowledgement Timeout

Test Objective	Rte_IFeedback API Functionality For Transmission Acknowledgement Timeout		
ID	ATS_RTE_00688	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01302 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531 RTE: SWS_Rte_03744 RTE: SWS_Rte_03746 RTE: SWS_Rte_04505 RTE: SWS_Rte_07367 RTE: SWS_Rte_07379 RTE: SWS_Rte_07647 RTE: SWS_Rte_07648		

	RTE: SWS_Rte_07650
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * a PPortPrototype which - sends VariableDataPrototype_TC51 typed by a primitive data type with swImplPolicy 'standard' - has a NonqueuedSenderComSpec with a TransmissionAcknowledgementRequest with timeout value > 0 * RunnableEntity_TC51 to initiate the test sequence - with a dataWriteAccess which references VariableDataPrototype_TC51 * RunnableEntity_TxTOut - with a dataWriteAccess which references VariableDataPrototype_TC51 - started by a DataWriteCompletedEvent which references this dataWriteAccess A communication matrix with * Signal_TC51 - mapped to VariableDataPrototype_TC51 * ISignalIPdu_TC51 - Signal_TC51 mapped to ISignalIPdu_TC51 with transferProperty triggered - configured with an EventControlledTiming - transmitted by the ECU
Summary	The test checks that after a failure to transmit data on time with Rte_IWrite a DataWriteCompletedEvent is triggered and the Rte_IFeedback provides access to the transmission error.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC51>] Invoke Rte_IWrite to write the dataElement VariableDataPrototype_TC51.	-
Step 2	-	[RUN<RunnableEntity_TxTOut>] RunnableEntity_TxTOut is invoked
Step 3	[RUN<RunnableEntity_TxTOut>] Invoke Rte_IFeedback for the dataElement VariableDataPrototype_TC51.	[RUN<RunnableEntity_TxTOut>] Rte_IFeedback returns RTE_E_TIMEOUT.
Post-conditions	NONE	

6.3.69 [ATS_RTE_00689] Data Invalidation For Implicit Communication

Test Objective	Data Invalidation For Implicit Communication		
ID	ATS_RTE_00689	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_01301 RTE: SWS_Rte_01302 RTE: SWS_Rte_02599 RTE: SWS_Rte_02600 RTE: SWS_Rte_02603 RTE: SWS_Rte_03741 RTE: SWS_Rte_03744 RTE: SWS_Rte_03746 RTE: SWS_Rte_03800 RTE: SWS_Rte_03801 RTE: SWS_Rte_06004 RTE: SWS_Rte_06011
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * A PPortPrototype to send - sends VariableDataPrototype_TC54 with swImplPolicy ‘standard’ - is typed by a SenderReceiverInterface with an invalidationPolicy for VariableDataPrototype_TC54 (handleInvalid = keep) * An RPortPrototype to receive VariableDataPrototype_TC54 - connected with the previous PPortPrototype * RunnableEntity_TC54 to execute the test sequence - with a dataReadAccess which references VariableDataPrototype_TC54 - with a dataWriteAccess which references VariableDataPrototype_TC54
Summary	The test invalidates a dataElement using the Rte_IInvalidate API and checks that the access to the status in another execution-instance using the Rte_IStatus API after a successful transmission with Rte_IWrite
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution

Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC54>] Invoke Rte_IWrite to write the dataElement VariableDataPrototype_TC54.	-
Step 2	[RUN<RunnableEntity_TC54>] Return to exit from this RunnableEntity_TC54 execution-instance.	-
Step 3	[CP] Wait for the next execution-instance of RunnableEntity_TC54.	[RUN<RunnableEntity_TC54>] RunnableEntity_TC54 is invoked.
Step 4	[RUN<RunnableEntity_TC54>] Invoke Rte_IInvalidate for the dataElement VariableDataPrototype_TC54.	-
Step 5	[RUN<RunnableEntity_TC54>] Invoke Rte_IStatus for the dataElement VariableDataPrototype_TC54.	[RUN<RunnableEntity_TC54>] Rte_IStatus returns RTE_E_OK.

Step 6	[RUN<RunnableEntity_TC54>] Return to exit from this RunnableEntity_TC54 execution-instance.	-
Step 7	[CP] Wait for the next execution-instance of RunnableEntity_TC54.	[RUN<RunnableEntity_TC54>] RunnableEntity_TC54 is invoked.
Step 8	[RUN<RunnableEntity_TC54>] Invoke Rte_IRead for the dataElement VariableDataPrototype_TC54.	[RUN<RunnableEntity_TC54>] Rte_IRead should read the configured invalid value.
Step 9	[RUN<RunnableEntity_TC54>] Invoke Rte_IStatus for the dataElement VariableDataPrototype_TC54.	[RUN<RunnableEntity_TC54>] Rte_IStatus returns RTE_E_INVALID.
Post-conditions	NONE	

6.3.70 [ATS_RTE_00690] Implicit Sender Receiver Communication On AliveTimeout

Test Objective	Implicit Sender Receiver Communication On AliveTimeout		
ID	ATS_RTE_00690	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01359 RTE: SWS_Rte_02599 RTE: SWS_Rte_02600 RTE: SWS_Rte_02604 RTE: SWS_Rte_03530 RTE: SWS_Rte_03531		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * an RPortPrototype which - receives VariableDataPrototype_TC55 typed by a primitive data type with swImplPolicy ‘standard’ - has a NonqueuedReceiverComSpec with aliveTimeout >0 * RunnableEntity_TC55AliveTimeOut - started by a DataReceiveErrorEvent which references VariableDataPrototype_TC55 - with a dataReadAccess which references VariableDataPrototype_TC55 A communication matrix with * Signal_TC55 - mapped to VariableDataPrototype_TC55 - received by the ECU		

Summary	This test case uses an inter-ECU communication where the communication is stopped for a longer time than the configured aliveTimeout. The test case verifies that this triggers a DataReceiveErrorEvent and that Rte_IStatus indicates a RTE_E_MAX_AGE_EXCEEDED status.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM channel shall be in COMM_FULL_COMMUNICATION mode LT shall transmit a frame with Signal_TC55 on the bus periodically

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[LT] Stop sending frame containing Signal_TC55 for more than the configured aliveTimeout.	[RUN<RunnableEntity_TC55AliveTimeOut>] RunnableEntity_TC55AliveTimeOut is invoked.
Step 2	[RUN<RunnableEntity_TC55AliveTimeOut>] Invoke Rte_IStatus for the dataElement VariableDataPrototype_TC55.	[RUN<RunnableEntity_TC55AliveTimeOut>] Rte_IStatus returns RTE_E_MAX_AGE_EXCEEDED.
Post-conditions	NONE	

6.3.71 [ATS_RTE_00695] Filter ‘never’ Configured

Test Objective	Filter ‘never’ Configured		
ID	ATS_RTE_00695	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_02516 RTE: SWS_Rte_02517 RTE: SWS_Rte_05500 RTE: SWS_Rte_05503 RTE: SWS_Rte_08077 RTE: SWS_Rte_08079		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which		

	- sends VariableDataPrototype_TC85 with swImplPolicy 'standard' * RPortPrototypeFilter which - receives VariableDataPrototype_TC85 with initValue 0x10 - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType=never * RPortPrototypeNoFilter which - receives VariableDataPrototype_TC85 with initValue 0x11 - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured without filter * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC85 - with a dataSendPoint which references VariableDataPrototype_TC85
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to never. The test checks that the data read is the initValue.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state
Main Test Execution	

Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Read_RPortPrototypeFilter_VariableDataPrototype_TC85.	[SWC] Rte_Read returns RTE_E_OK and provides the initValue 0x10.
Step 2	[SWC] Invoke Rte_Read_RPortPrototypeNoFilter_VariableDataPrototype_TC85.	[SWC] Rte_Read returns RTE_E_OK and provides the initValue 0x11.
Step 3	[SWC] Invoke Rte_Write with 0x55.	[SWC] Rte_Write returns RTE_E_OK.
Step 4	[SWC] Invoke Rte_Read_RPortPrototypeFilter_VariableDataPrototype_TC85.	[SWC] Rte_Read returns RTE_E_OK and provides the initValue 0x10.
Step 5	[SWC] Invoke Rte_Read_RPortPrototypeNoFilter_VariableDataPrototype_TC85.	[SWC] Rte_Read returns RTE_E_OK and provides the value 0x55.
Post-conditions	NONE	

6.3.72 [ATS_RTE_00696] Filter 'maskedNewEqualsX' Configured

Test Objective	Filter 'maskedNewEqualsX' Configured		
ID	ATS_RTE_00696	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2

Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC86 with swImplPolicy ‘standard’ * An RPortPrototype which - receives VariableDataPrototype_TC86 with initValue (Ex: 0x10) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType=maskedNewEqualsX mask=Ex. 0x0F x=Ex. 0x00 * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC86 - with a dataSendPoint which references VariableDataPrototype_TC86 User defined data: Data_1 = Ex: 0x55 (does not match the filter) Data_2 = Ex: 0x50 (matches the filter)		
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to maskedNewEqualsX. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	[SWC] Invoke Rte_Write to write Data_1.	[SWC] Rte_Write returns RTE_E_OK.	
Step 2	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides the InitValue.	
Step 3	[SWC] Invoke Rte_Write to write Data_2.	[SWC] Rte_Write returns RTE_E_OK.	
Step 4	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.	
Step 5	[SWC] Invoke Rte_Write to write Data_1.	[SWC] Rte_Write returns RTE_E_OK.	
Step 6	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.	
Post-conditions	NONE		

6.3.73 [ATS_RTE_00697] Filter 'maskedNewDiffersX' Configured

Test Objective	Filter 'maskedNewDiffersX' Configured		
ID	ATS_RTE_00697	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC87 with swImplPolicy 'standard' * An RPortPrototype which - receives VariableDataPrototype_TC87 with initValue (Ex: 0x10) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType=maskedNewDiffersX mask=Ex. 0x0F x=Ex. 0x05 * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC87 - with a dataSendPoint which references VariableDataPrototype_TC87 User defined data:Data_1 = Ex: 0x55 (does not match the filter) Data_2 = Ex: 0x50 (matches the filter)		
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to maskedNewDiffersX. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write Data_1.		[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Read.		[SWC] Rte_Read returns RTE_E_OK and provides the initValue.
Step 3	[SWC] Invoke Rte_Write to write Data_2.		[SWC] Rte_Write returns RTE_E_OK.
Step 4	[SWC] Invoke Rte_Read.		[SWC] Rte_Read returns RTE_E_OK and provides Data_2.

Post-conditions	NONE

6.3.74 [ATS_RTE_00698] Filter 'maskedNewDiffersMaskedOld' Configured

Test Objective	Filter 'maskedNewDiffersMaskedOld' Configured		
ID	ATS_RTE_00698	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC88 with swImplPolicy 'standard' * An RPortPrototype which - receives VariableDataPrototype_TC88 with initValue (0x11) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType= maskedNewDiffersMaskedOld mask=Ex. 0x0F * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC88 - with a dataSendPoint which references VariableDataPrototype_TC88		
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to maskedNewDiffersMaskedOld. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match. Another data is sent at the end that does not match the filter but would have matched in case the previous value was accepted.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write 0x21.		[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Read.		[SWC] Rte_Read returns RTE_E_OK and provides the value 0x11.
Step 3	[SWC] Invoke Rte_Write to write 0x22.		[SWC] Rte_Write returns RTE_E_OK.

Step 4	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides the value 0x22.
Step 5	[SWC] Invoke Rte_Write to write 0x32.	[SWC] Rte_Write returns RTE_E_OK.
Step 6	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides the value 0x22.
Post-conditions	NONE	

6.3.75 [ATS_RTE_00699] Filter 'newIsWithin' Configured

Test Objective	Filter 'newIsWithin' Configured		
ID	ATS_RTE_00699	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC89 with swImplPolicy 'standard' * An RPortPrototype which - receives VariableDataPrototype_TC89 with initValue (Ex: 0x10) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType=newIsWithin min=Ex. 0x10 max=Ex. 0x50 * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC89 - with a dataSendPoint which references VariableDataPrototype_TC89 User defined data: Data_1 = Ex: 0x64 (does not match the filter) Data_2 = Ex: 0x15 (matches the filter) Data_3 = Ex: 0x10 (matches the filter – min value) Data_4 = Ex: 0x50 (matches the filter – max value)		
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to newIsWithin. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match.		
Needed Adaptation to other Releases			

Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write Data_1.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides the initValue.
Step 3	[SWC] Invoke Rte_Write to write Data_2.	[SWC] Rte_Write returns RTE_E_OK.
Step 4	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.
Step 5	[SWC] Invoke Rte_Write to write Data_3.	[SWC] Rte_Write returns RTE_E_OK.
Step 6	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_3.
Step 7	[SWC] Invoke Rte_Write to write Data_4.	[SWC] Rte_Write returns RTE_E_OK.
Step 8	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_4.
Post-conditions	NONE	

6.3.76 [ATS_RTE_00700] Filter ‘newIsOutside’ Configured

Test Objective	Filter ‘newIsOutside’ Configured		
ID	ATS_RTE_00700	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC90 with swImplPolicy ‘standard’ * An RPortPrototype which - receives VariableDataPrototype_TC90 with initValue (Ex: 0x09) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter:		

	dataFilterType=newIsOutside min=Ex. 0x10 max=Ex. 0x50 * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC90 - with a dataSendPoint which references VariableDataPrototype_TC90 User defined data: Data_1 = Ex: 0x15 (does not match the filter) Data_2 = Ex: 0x64 (matches the filter) Data_1 = Ex: 0x10 (does not match the filter – min value) Data_2 = Ex: 0x50 (does not match the filter – max value)
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to newIsOutside. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write Data_1.	[SWC] Rte_Write return RTE_E_OK.
Step 2	[SWC] Invoke Rte_Read.	[SWC] Rte_Read return RTE_E_OK and provides the initValue.
Step 3	[SWC] Invoke Rte_Write to send Data_2.	[SWC] Rte_Write return RTE_E_OK.
Step 4	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.
Step 5	[SWC] Invoke Rte_Write to send Data_3.	[SWC] Rte_Write return RTE_E_OK.
Step 6	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.
Step 7	[SWC] Invoke Rte_Write to send Data_4.	[SWC] Rte_Write return RTE_E_OK.
Step 8	[SWC] Invoke Rte_Read.	[SWC] Rte_Read returns RTE_E_OK and provides Data_2.
Post-conditions	NONE	

6.3.77 [ATS_RTE_00701] Filter ‘oneEveryN’ Configured

Test Objective	Filter ‘oneEveryN’ Configured		
ID	ATS_RTE_00701	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement			

on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_01091 RTE: SWS_Rte_01093 RTE: SWS_Rte_01289 RTE: SWS_Rte_05503
Requirements / Reference to Test Environment	
Configuration Parameters	A SW-C with * A PPortPrototype which - sends VariableDataPrototype_TC91 with swImplPolicy 'standard' * An RPortPrototype which - receives VariableDataPrototype_TC91 with initValue (Ex: 0x09) - is connected with the previous PPortPrototype - has a NonqueuedReceiverComSpec configured with filter: dataFilterType=oneEveryN offset=2 period=3 * A RunnableEntity to execute the test sequence - with a dataReceivePointByArgument which references VariableDataPrototype_TC91 - with a dataSendPoint which references VariableDataPrototype_TC91
Summary	The test writes some data on a dataElement and reads it on a port with a filter set to oneEveryN. The test checks that the reception filter was correctly applied by sending one value which match the filter and another one which does not match.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Write to write value 1.	[SWC] Rte_Write returns RTE_E_OK.
Step 2	[SWC] Invoke Rte_Read	[SWC] Rte_Read returns RTE_E_OK and provides the initValue.
Step 3	[SWC] Invoke Rte_Write to write value 2.	[SWC] Rte_Write returns RTE_E_OK.
Step 4	[SWC] Invoke Rte_Read	[SWC] Rte_Read returns RTE_E_OK and provides the value 2.
Step 5	[SWC] Invoke Rte_Write to write value 3.	[SWC] Rte_Write returns RTE_E_OK.
Step 6	[SWC] Invoke Rte_Read	[SWC] Rte_Read returns RTE_E_OK and provides the value 2.
Step 7	[SWC] Invoke Rte_Write to write value 4.	[SWC] Rte_Write returns RTE_E_OK.
Step 8	[SWC] Invoke Rte_Read	[SWC] Rte_Read returns RTE_E_OK and provides the value 2.
Step 9	[SWC] Invoke Rte_Write to write value 5.	[SWC] Rte_Write returns RTE_E_OK.

Step 10	[SWC] Invoke Rte_Read	[SWC] Rte_Read returns RTE_E_OK and provides the value 5.
Post-conditions	NONE	

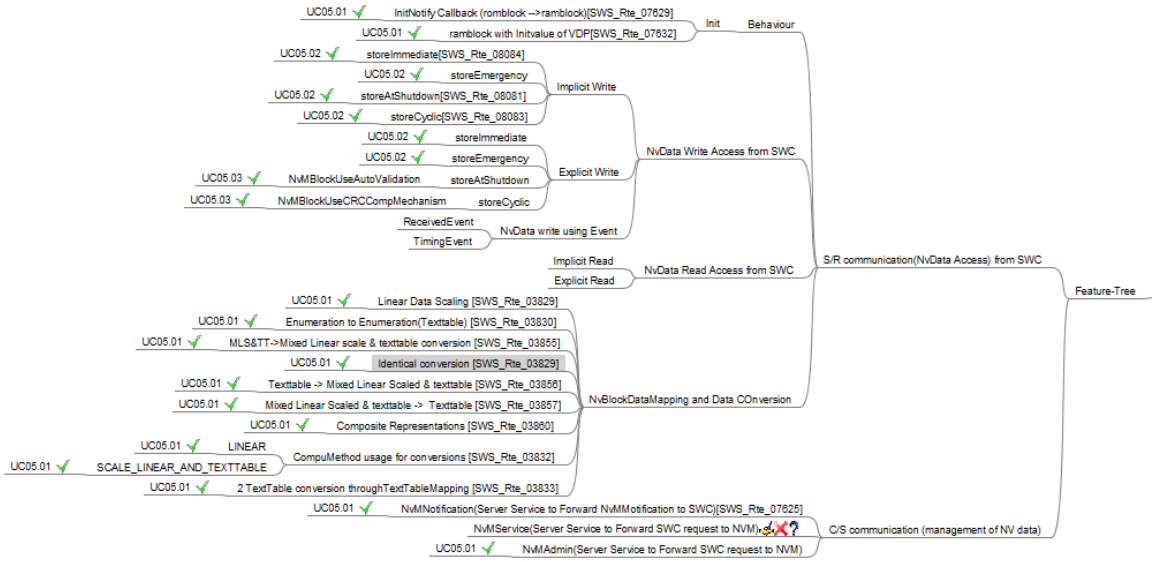
7 RS_BRF_01416_NvDataHandling

7.1 General Test Objective and Approach

This Test Specification covers the NvDataHandling via RTE as described in the AUTOSAR Feature [RS_BRF_01416].

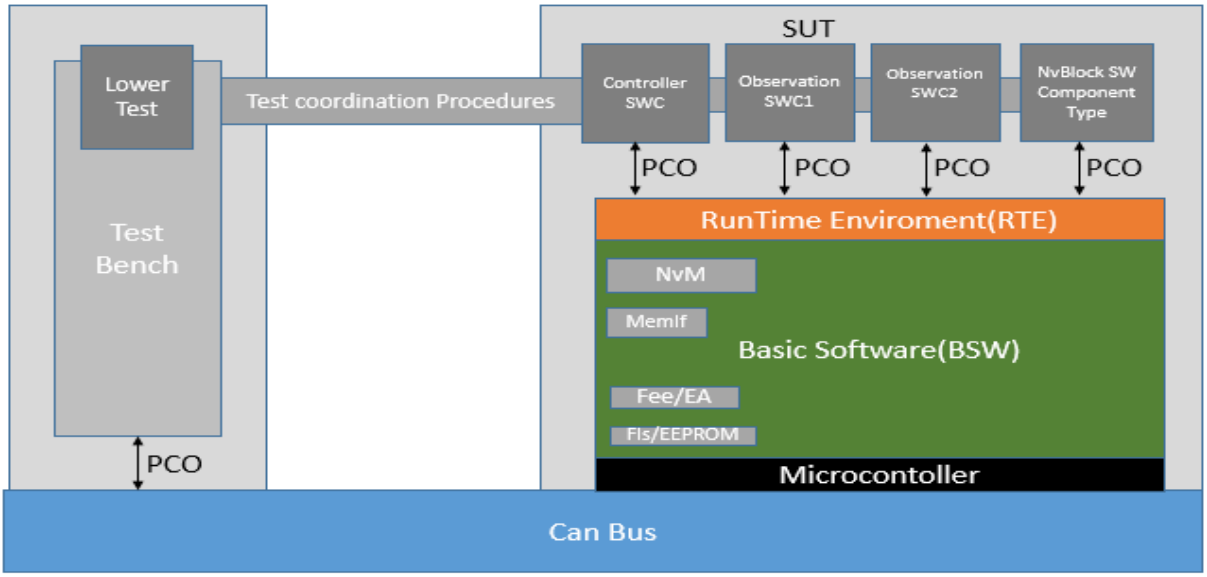
The tests use a test bench environment and embedded Software Components that use the feature.

This test specification document has been established to cover the following features:



7.1.1 Test System

7.1.1.1 Overview on Architecture



The test system architecture requires three SWC's as upper testers and one NvBlockSoftwareComponent.

One is the Control_SWC and other two are Observation_SWC1 and Observation_SWC2. Observation_SWC1 and Observation_SWC2 is used to read NvBlockSoftwareComponent data elements to validate the data received in NvBlockSoftwareComponent from Control_SWC.

Four NvBlockDescriptor is configured in NvBlockSoftwareComponent for different Writing Strategies.

The test cases are derived based on below use cases.

UC05.01 : NvData Access from SWC

Writing of data to Nv Block and Reading it from Nv Block for different data types from SWC both implicitly and explicitly.

UC05.02 : NvData Writing Strategies

Four Different Writing Strategies are there to write NV Block.

1. StoreCyclic : NvM_WriteBlock will be invoked in next periodic activity when it is configured as True for respective Block Descriptor.
2. StoreImmediate : NvM_WriteBlock will be invoked by Rte immediately when it is configured as True for respective Block Descriptor.
3. StoreAtShudown : Rte will invoke NvM_SetRamBlockStatus for respective Block Descriptor. NvM_WriteAll will be called by EcuM at the time of shutdown to write Nv Blocks
4. StoreEmergency : CDD will invoke NvM_WriteBlock for respective Block Descriptor. This scenario is not handled in RTE and it is not tested in ATS.

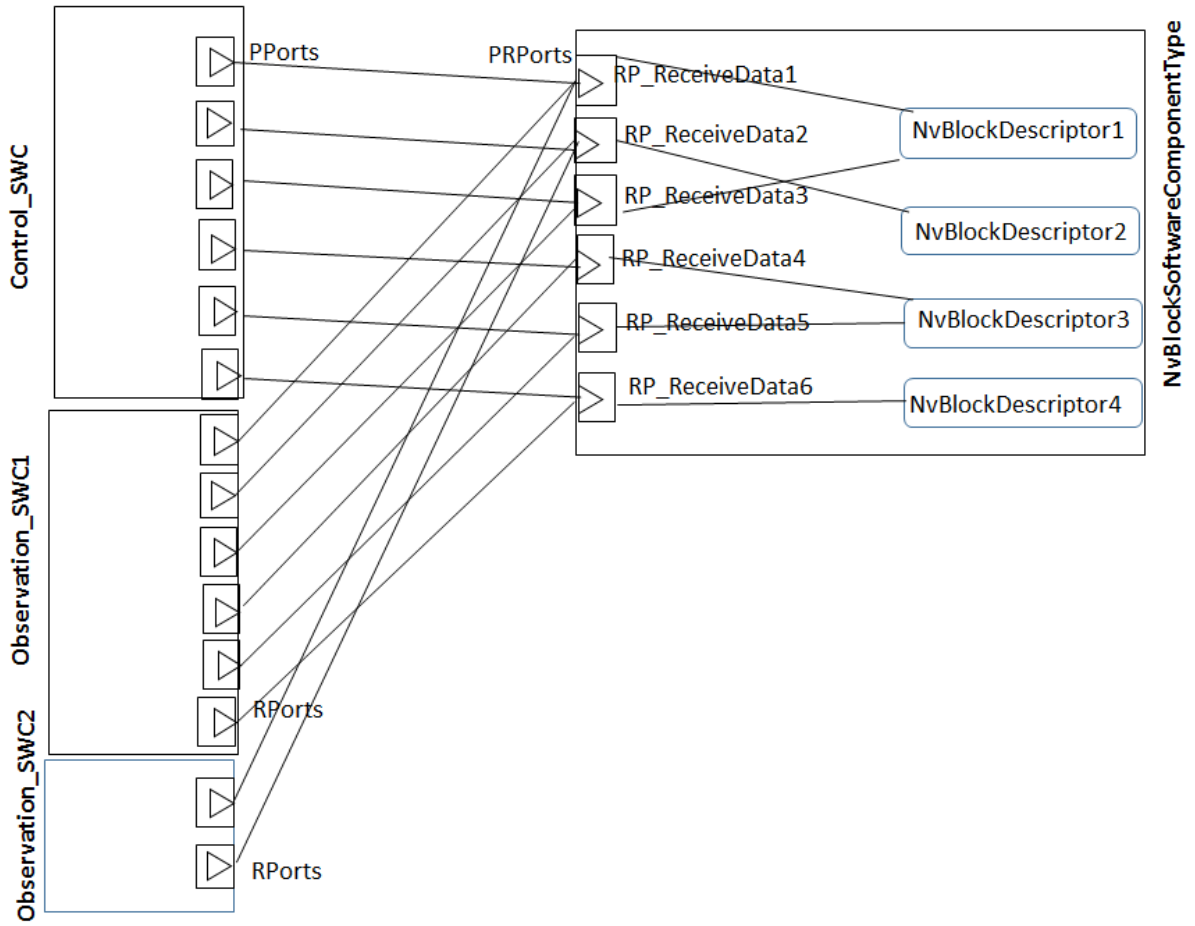
Uc05.03 : Optimization of Write Operations to Nv Memory

If data is not changed from previous Nv Block Write, NvM will not invoke Write to the lower layers. The below parameters should be configured as True.

1. calcRamBlockCrc should be True
2. useCRCCompMechanism should be True

Sender/Receiver Communication

S/R Communication is used for data conversion from Control_SWC to NvBlockSwComponent

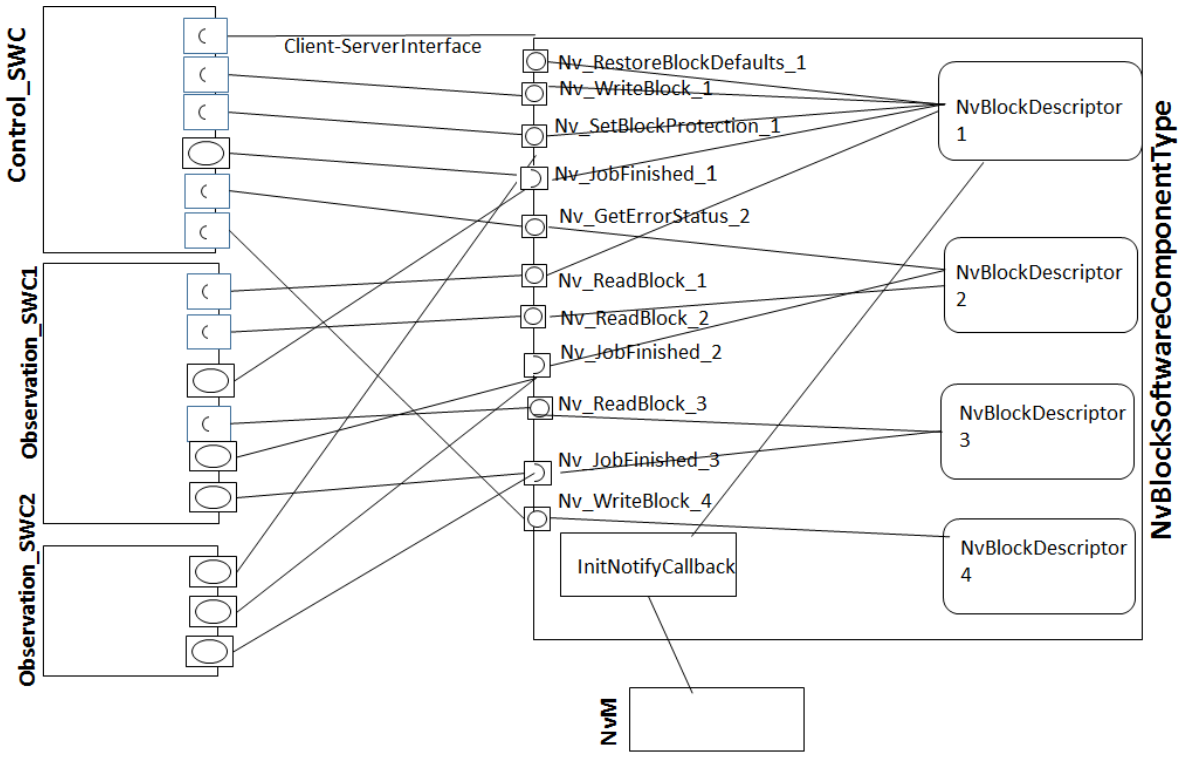


Client-Server Communication

JobEndNotification from NvM is indicated to Control_SWC, Observation_SWC1 and Observation_SWC2 via C/S Interface.

Observation_SWC1 will invoke NvM_ReadBlock() for NvBlockDescriptor1, NvBlockDescriptor2, NvBlockDescriptor3 which is configured as Immediate, Cyclic, Cyclic and Shutdown, to read the data written in Nv Block via C/S Interface.

Control_SWC will invoke NvM_WriteBlock() for NvBlockDescriptor4 which is not configured for any of the writing strategies and Observation_SWC1 will invoke NvM_ReadBlock() for the same.



7.1.1.2 Specific Requirements

NvBlockDescriptor should be configured in NvBlockSoftwareComponentType to provide RAM Block to SWC and the data copied in RAM Block should be stored in NVRAM Block based on different Writing Strategies.

7.1.1.3 Test Coordination Requirements

Test Coordination Procedures are needed to synchronize the runnables of the SWC and of the NvBlockSoftwareComponentType in test cases.

TCPs are also needed to collect the test results of the SWCs and the Bus simulation at one central place in order to derive the test verdict.

It is up to the test system designer/implementer to define that “central place” and to design/implement the test coordination functionality.

7.1.2 Test Configuration

Key Configuration Parameters details are provided in individual testcases. .

7.1.2.1 Required ECU Extract of System Description Files

None.

7.1.2.2 Required ECU Configuration Description Files

The NVRAM block associated to NvBlockDescriptors needs to be configured with below values.

NvMBlockUseSyncMechanism should be configured as True.

NvMWriteRamBlockToNvCallback and NvMReadRamBlockFromNvCallback should be configured.

7.1.2.3 Required Software Component Description Files

7.1.2.3.1 SWC Control_SWC

SWC Name	Control_SWC	
PORTS	**Name**	PP_SendData1
	Type	PPortPrototype
	Interface	IF_ExchangedData1
	ComSpecs	NvProvideComSpec for dataElement VDP_Nv1
	Requirements	Compu Method : Identical
	Name	PP_SendData2
	Type	PPortPrototype
	Interface	IF_ExchangedData2
	ComSpecs	NvProvideComSpec for Array VDP_Nv2
	Requirements	Compu Method : Identical
	Name	PP_SendData3
	Type	PPortPrototype
	Interface	IF_ExchangedData3

	ComSpecs	NvProvideComSpec for Structure
		VDP_ImplWrite2
		Record
		{
		App_Data1 : uint16
		App_Data2 : uint16
		App_Data3 : uint16
		App_Data4 : uint16
		}

	Requirements	
	Name	PP_SendData4
	Type	PPortPrototype
	Interface	IF_ExchangedData4
	ComSpecs	NvProvideComSpec for Boolean Element VDP_ExplWrite2 App_Boolean_Data
	Requirements	Compu Method : TEXTTABLE

	Name	PP_SendData5		
	Type	PPortPrototype		
	Interface	IF_ExchangedData5		
	ComSpecs	NvProvideComSpec for data Element VDP_Nv5		
	Requirements	Compu Method : Identical		
	Name	PP_SendData6		
	Type	PPortPrototype		
	Interface	IF_ExchangedData5		
	ComSpecs	NvProvideComSpec for data Element VDP_Nv6		
	Requirements	Compu Method : Identical		
RUNNABLE ENTITIES	**Name**	Run_Ctrl_1		
	Requirements	Executed in the same task as Run_Obs1_1 and Run_Obs2_1		
	VariableDataPrototype	**Name**	VDP_Nv1	
		Type	DataWriteAccess1	
		Access to	PP_SendData1/ IF_ExchangedData1/ App_Data5	
		Requirements	Belongs to the same coherency group as Run_Obs1_1/ VDP_Nv1 and Run_Obs2_1/VDP_Nv1 , Compu Method : Identical	
	Name	Run_Ctrl_2		
	Requirements	Executed in the same task as Run_Obs1_2 and Run_Obs2_2		
	VariableDataPrototype	**Name**	VDP_Nv2	
		Type	DataSendPoint1	
		Access to	PP_SendData2/ IF_ExchangedData2/ App_Array[4] : uint16	
		Requirements	Belongs to the same coherency group as Run_Obs1_2/ VDP_Nv2 and Run_Obs2_2/VDP_Nv2 Compu Method : Identical	

	Name	Run_Ctrl_3	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_ImplWrite2
		Type	DataWriteAccess2
		Access to	PP_SendData3/ IF_ExchangedData3/ Record{App_Data1 : uint16 App_Data2 : uint16 App_Data3 : uint16 App_Data4 : uint16 }
		Requirements	None
	Name	Run_Ctrl_4	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_ExplWrite2
		Type	DataSendPoint2
		Access to	PP_SendData4/ IF_ExchangedData4/ App_Boolean_Data
		Requirements	Compu Method : TEXTTABLE
	Name	Run_Ctrl_5	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv5
		Type	DataSendPoint3
		Access to	PP_SendData5/ IF_ExchangedData5/ App_Data6 : uint16
		Requirements	Compu Method : Identical
	Name	Run_Ctrl_6	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv6
		Type	DataSendPoint4
		Access to	PP_SendData6/ IF_ExchangedData6/ App_Data7 : uint16
		Requirements	Compu Method : Identical

7.1.2.3.2 SWC NvBlockSoftwareComponentType

SWC Name	NvBlockSoftwareComponentType	
NvBlockDescriptor	**Name**	NvBlockDescriptor1
	Requirements	NvBlockNeeds supportDirtyFlag – True storeImmediate - True
	Name	NvBlockDataMapping1
	ComSpecs	Refer PR_ReceiveData1
	Requirements	WrittenReadNvData, NvRamBlockElement: Nv_Data4
	Name	NvBlockDataMapping3
	ComSpecs	Refer PR_ReceiveData3
	Requirements	WrittenReadNvData, NvRamBlockElement; Nv_Record{ Nv_Data1 : uint16 Nv_Data2 : uint16 }
	Name	NvBlockDescriptor2
	Requirements	NvBlockNeeds supportDirtyFlag – True storeCyclic – True cyclicWritingPeriod – 10seconds useCRCCompMechanism - True
	Name	NvBlockDataMapping2
	ComSpecs	Refer PR_ReceiveData2
	Requirements	WrittenReadNvData, NvRamBlockElement; Nv_Array[4]
	Name	NvBlockDescriptor3
	Requirements	NvBlockNeeds supportDirtyFlag – True storeCyclic – True cyclicWritingPeriod – 10 seconds storeAtShutdown – True useAutoValidationAtShudown - True
	Name	NvBlockDataMapping4
	ComSpecs	Refer PR_ReceiveData4
	Requirements	WrittenReadNvData, NvRamBlockElement: Nv_Data3_Bitfield : uint8

	Name	NvBlockDataMapping5
	ComSpecs	Refer PR_ReceiveData5
	Requirements	WrittenReadNvData, NvRamBlockElement Nv_Data5
	Name	NvBlockDescriptor4
	Requirements	supportDirtyFlag – False
	Name	NvBlockDataMapping6
	ComSpecs	Refer PR_ReceiveData6
	Requirements	WrittenReadNvData, NvRamBlockElement Nv_Data6
PORTS	**Name**	PR_ReceiveData1
	Type	PRPortPrototype
	Interface	IF_ExchangedData1
	ComSpecs	NvProvideComSpec and NvRequireComSpec for data element VDP_Nv1 : App_Data5 (ApplicationPrimitiveDataType)
	Requirements	Compu Method : Identical
	Name	PR_ReceiveData1Event
	Type	Data Received Event
	Reference	Start on Event, Target Data Element
	Name	PR_ReceiveData2
	Type	PRPortPrototype
	Interface	IF_ExchangedData2
	ComSpecs	NvProvideComSpec and NvRequireComSpec for Array VDP_Nv2 : App_Array[4] (ApplicationArrayDataType)
	Requirements	Compu Method : Identical
	Name	PR_TimingData2Event
	Type	Timing Event
	Reference	Start on Event, Event Reference, Timing Period
	Name	PR_ReceiveData3

	Type	PRPortPrototype
	Interface	IF_ExchangedData3
	ComSpecs	NvProvideComSpec and NvRequireComSpec for Structure Nv_Record { Nv_Data1 : uint16 Nv_Data2 : uint16 }
	Requirements	
	Name	PR_ReceiveData3Event
	Type	Data Received Event
	Reference	Start on Event, Target Data Element
	Name	PR_ReceiveData4
	Type	PRPortPrototype
	Interface	IF_ExchangedData4
	ComSpecs	NvProvideComSpec and NvRequireComSpec for Bitfield Nv_Data3_Bitfield : uint8
	Requirements	Compu Method : TEXTTABLE Mapping : 3rd bit needs to be set/reset in Nv_data3_Bitfield.
	Name	PR_TimingData4Event
	Type	Timing Event
	Reference	Start on Event, Event Reference, Timing Period
	Name	PR_ReceiveData5
	Type	PRPortPrototype
	Interface	IF_ExchangedData5
	ComSpecs	NvProvideComSpec and NvRequireComSpec for Primitive Data Type VDP_Nv5 : App_Data6 (ApplicationPrimitiveDataType)
	Requirements	Compu Method : Identical
	Name	PR_TimingData5Event
	Type	Timing Event
	Reference	Start on Event, Event Reference, Timing Reference

	Name	PR_ReceiveData6	
	Type	PRPortPrototype	
	Interface	IF_ExchangedData6	
	ComSpecs	NvProvideComSpec and NvRequireComSpec for Primitive Data Type VDP_Nv6 : App_Data7 (ApplicationPrimitiveDataType)	
	Requirements	Compu Method : Identical	
RUNNABLE ENTITIES	**Name**	Run_NvM_WriteBlock	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv1, VDP_Nv3
		VDP InitValue	VDP_Nv1 = 0x190
		Type	
		Access to	PR_ReceiveData1/ IF_ExchangedData1/ Nv_Data4, PR_ReceiveData3/ IF_ExchangedData3/ Nv_Record
		Requirements	DataTypeMapping : App_Data5 : uint32 Record {App_Data1 : uint16 App_Data2 : uint16 App_Data3 : uint16 App_Data4 : uint16
	Name	Run_NvM_WriteBlock_2	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv2
		Type	
		Access to	PR_ReceiveData2/ IF_ExchangedData2/ Nv_Array,
		Requirements	DataTypeMapping : App_Array[4] : uint16
	Name	Run_NvM_WriteBlock_3	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv4

		Type	
		Access to	PR_ReceiveData4/ IF_ExchangedData4/ Nv_Data3_Bitfield
		Requirements	
	Name	Run_WriteBlock_5	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv5
		Type	
		Access to	PR_ReceiveData5/ IF_ExchangedData5/ Nv_Data5
		Requirements	

7.1.2.3.3 SWC Observation_SWC1

SWC Name	Observation_SWC1	
PORTS	**Name**	RP_ObserveData1
	Type	RPortPrototype
	Interface	IF_ExchangedData1
	ComSpecs	NvRequireComSpec for data element Nv_Data4 : uint32
	Name	RP_ObserveData2
	Type	RPortPrototype
	Interface	IF_ExchangedData2
	ComSpecs	NvRequireComSpec for Array Nv_Array[4] : uint16
	Name	RP_ObserveData3
	Type	RPortPrototype
	Interface	IF_ExchangedData3
	ComSpecs	NvRequireComSpec for Structure Nv_Record { Nv_Data1 : uint16 Nv_Data2 : uint16 }
	Name	RP_ObserveData4
	Type	RPortPrototype
	Interface	IF_ExchangedData4
	ComSpecs	NvRequireComSpec for Bitfield

		Nv_Data3_Bitfield : uint8	
	Requirements	Compu Method : TEXTTABLE Mapping : App_Boolean_Data is set/reset 3rd bit of Nv_Data3_Bitfield in RAMBlock	
	Name	RP_ObserveData5	
	Type	RPortPrototype	
	Interface	IF_ExchangedData5	
	ComSpecs	NvRequireComSpec for data Element Nv_Data5 : uint16	
	Requirements	Compu Method : Identical	
	Name	RP_ObserveData8	
	Type	RPortPrototype	
	Interface	IF_ExchangedData6	
	ComSpecs	NvRequireComSpec for data Element Nv_Data6 : uint16	
	Requirements	Compu Method : Identical	
RUNNABLE ENTITIES	**Name**	Run_Obs1_1	
	Requirements	Executed in the same task as Run_Ctrl_1 and Run_Obs2_1	
	VariableDataPrototype	**Name**	VDP_Nv1
		Type	DataReadAccess1
		Access to	RP_ObserveData1/ IF_ExchangedData1/ Nv_Data4
		Requirements	Belongs to the same coherency group as Run_Ctrl_1 and Run_Obs2_1
	Name	Run_Obs1_2	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv2
		Type	DataReceivePoint1
		Access to	RP_ObserveData2/ IF_ExchangedData2/ Nv_Array[4]
		Requirements	

	Name	Run_Obs1_3	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv3
		Type	DataReadAccess2
		Access to	RP_ObserveData3/ IF_ExchangedData3/ Nv_Record
		Requirements	Belongs to the same coherency group as Run_Ctrl_3 and Run_Obs2_3/
	Name	Run_Obs1_4	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv4
		Type	DataReceivePoint2
		Access to	RP_ObserveData4/ IF_ExchangedData4/ Bitfield data Nv_Data3_Bitfield
		Requirements	
	Name	Run_Obs1_5	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv5
		Type	DataReceivePoint3
		Access to	RP_ObserveData5/ IF_ExchangedData5/ Nv_Data5
		Requirements	
	Name	Run_Obs1_6	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv6
		Type	DataReceivePoint3
		Access to	RP_ObserveData5/ IF_ExchangedData5/ Nv_Data5
		Requirements	

7.1.2.3.4 SWC Observation_SWC2

SWC Name	Observation_SWC2

PORTS	**Name**	RP_ObserveData6	
	Type	RPortPrototype	
	Interface	IF_ExchangedData1	
	ComSpecs	NvRequireComSpec for data element Nv_Data4	
	Requirements	None	
	Name	RP_ObserveData7	
	Type	RPortPrototype	
	Interface	IF_ExchangedData2	
	ComSpecs	NvRequireComSpec for ArrayElement Nv_Array[4]	
	Requirements	None	
RUNNABLE ENTITIES	**Name**	Run_Obs2_1	
	Requirements	Executed in the same task as Run_Ctrl_1 and Run_Obs1_1	
	VariableDataPrototype	**Name**	VDP_Nv1
		Type	DataReadAccess1
		Access to	RP_ObserveData6/ IF_ExchangedData2/ Nv_Data4
		Requirements	Belongs to the same coherency group as Run_Ctrl_1 and Run_Obs1_1
	Name	Run_Obs2_2	
	Requirements	None	
	VariableDataPrototype	**Name**	VDP_Nv2
		Type	DataReceivePoint1
		Access to	RP_ObserveData7/ IF_ExchangedData2/ Nv_Array[4]
		Requirements	Belongs to the same coherency group as Run_Ctrl_2 and Run_Obs1_2

SWC Name	Control_SWC			
PORTS	**Name**	Ctrl_RestoreBlockDefaults_Client_1		
	Type	RPortPrototype		
	Interface	RestoreBlockDefaults		
	Name	Ctrl_JobFinished_1		
	Type	PPortPrototype		
	Interface	JobFinished		
	Name	Ctrl_WriteBlock_1		
	Type	RPortPrototype		
	Interface	WriteBlock		
	Name	Ctrl_SetBlockProtection_1		
	Type	RPortPrototype		
	Interface	SetBlockProtection		
	Name	Ctrl_GetErrorStatus_2		
	Type	RPortPrototype		
	Interface	GetErrorStatus		
	Name	Ctrl_WriteBlock_4		
	Type	RPortPrototype		
	Interface	WriteBlock_4		
RUNNABLE ENTITIES	**Name**	Run_Ctrl_RestoreBlockDefaults_1		
	Requirements	None		
	Servercall points	**Name**	sscp_RestoreBlock	
		Type	SynchronousServerCallPoint	
		Access to	Ctrl_RestoreBlockDefaults_Client_1	
		Requirements	None	
	Name	Run_Ctrl_JobFinished		
	Requirements	None		
		Access to	Ctrl_JobFinished_1	
		Requirements	None	
	Name	Run_Ctrl_WriteBlock_1		

	Requirements	None	
	Servercall points	**Name**	sscp_WriteBlock
		Type	SynchronousServerCallPoint
		Access to	Ctrl_WriteBlock_1
		Requirements	None
	Name	Run_Ctrl_SetBlockProtection_1	
	Requirements	None	
	Servercall points	**Name**	sscp_SetBlockProtection
		Type	SynchronousServerCallPoint
		Access to	Ctrl_SetBlockProtection_1
		Requirements	None
	Name	Run_Ctrl_GetErrorStatus_2	
	Requirements	None	
	Servercall points	**Name**	sscp_GetErrorStatus
		Type	SynchronousServerCallPoint
		Access to	Ctrl_GetErrorStatus_2
		Requirements	None
	Name	Run_Ctrl_WriteBlock_4	
	Requirements	None	
	Servercall points	**Name**	sscp_WriteBlock
		Type	SynchronousServerCallPoint
		Access to	Ctrl_WriteBlock_4
		Requirements	None

SWC Name	NvBlockSwComponentType	
PORTS	**Name**	Nv_RestoreBlockDefaults_1
	Type	PPortPrototype
	Interface	Nv_RestoreBlockDefaults_1
	Name	Nv_JobFinished_1
	Type	RPortPrototype
	Interface	Nv_JobFinished_1
	Name	Nv_JobFinished_1_Obs1

	Type	RPortPrototype
	Interface	Nv_JobFinished_1_Obs1
	Name	Nv_WriteBlock_1
	Type	PPortPrototype
	Interface	Nv_WriteBlock_1
	Name	Nv_SetBlockProtection_1
	Type	PPortPrototype
	Interface	Nv_SetBlockProtection_1
	Name	Nv_ReadBlock_1
	Type	PPortPrototype
	Interface	Nv_ReadBlock_1
	Name	Nv_ReadBlock_2
	Type	PPortPrototype
	Interface	Nv_ReadBlock_2
	Name	Nv_ReadBlock_3
	Type	PPortPrototype
	Interface	Nv_ReadBlock_3
	Name	Nv_JobFinished_2
	Type	RPortPrototype
	Interface	Nv_JobFinished_2
	Name	Nv_JobFinished_3
	Type	RPortPrototype
	Interface	Nv_JobFinished_3
	Name	Nv_GetErrorStatus_2
	Type	PPortPrototype
	Interface	Nv_GetErrorStatus_2
	Name	Nv_WriteBlock_4
	Type	PPortPrototype
	Interface	Nv_WriteBlock_4

	Name	Nv_JobFinished_4_Obs1	
	Type	RPortPrototype	
	Interface	Nv_JobFinished_4_Obs1	
Operation Invoked Event	**Name**	OperationInvokeEvent_RestoreBlockDefaults	
	Runnable Entity	Run_Nv_RestoreBlockDefaults_1	
		Access to	Nv_RestoreBlockDefaults_1
		Requirements	None
	Name	OperationInvokedEvent_Nvm_SetBlockProtection	
	Runnable Entity	Run_Nv_SetBlockProtection_1	
		Access to	Nv_SetBlockProtection_1
		Requirements	None
	Name	OperationInvokeEvent_WriteBlock	
	Runnable Entity	Run_Nv_WriteBlock_1	
		Access to	Nv_WriteBlock_1
		Requirements	None
	Name	OperationInvokeEvent_ReadBlock_1	
	Runnable Entity	Run_Nv_ReadBlock_1	
		Access to	Nv_ReadBlock_1
		Requirements	None
	Name	OperationInvokeEvent_ReadBlock_2	
	Runnable Entity	Run_Nv_ReadBlock_2	
		Access to	Nv_ReadBlock_2
		Requirements	None
	Name	OperationInvokeEvent_ReadBlock_3	
	Runnable Entity	Run_Nv_ReadBlock_3	
		Access to	Nv_ReadBlock_3
		Requirements	None
	Name	OperationInvokeEvent_GetErrorStatus_2	
	Runnable Entity	Run_Nv_GetErrorStatus_2	
		Access to	Nv_GetErrorStatus_2

		Requirements	None
	Name	OperationInvokeEvent_WriteBlock_4	
	Runnable Entity	Run_Nv_WriteBlock_4	
		Access to	Nv_WriteBlock_4
		Requirements	None

SWC Name	Observation_SWC1	
PORTS	**Name**	Obs1_ReadBlock_1
	Type	RPortPrototype
	Interface	ReadBlock_1
	Name	Obs1_ReadBlock_2
	Type	RPortPrototype
	Interface	ReadBlock_2
	Name	Obs1_ReadBlock_3
	Type	RPortPrototype
	Interface	ReadBlock_1
	Name	Obs1_JobFinished_1
	Type	PPortPrototype
	Interface	JobFinished_1
	Name	Obs1_JobFinished_2
	Type	PPortPrototype
	Interface	JobFinished_2
	Name	Obs1_JobFinished_3
	Type	PPortPrototype
	Interface	JobFinished_3
	Name	Obs1_JobFinished_4
	Type	PPortPrototype
	Interface	JobFinished_4
Operation Invoked Event	**Name**	OperationInvokedEvent_Obs
	Runnable Entity	Run_Obs1_JobFinished_1

		Access to	Obs1_JobFinished_1
		Requirements	None
	Name	OperationInvokedEvent_Obs1_2	
	Runnable Entity	Run_Obs1_JobFinished_2	
		Access to	Obs1_JobFinished_2
		Requirements	None
	Name	OperationInvokedEvent_Obs1_3	
	Runnable Entity	Run_Obs1_JobFinished_3	
		Access to	Obs1_JobFinished_3
		Requirements	None
RUNNABLE ENTITIES	**Name**	Run_Obs1_ReadBlock_1	
	Requirements	None	
	Servercall points	**Name**	sscp_ReadBlock_1
		Type	SynchronousServerCallPoint
		Access to	Obs1_ReadBlock_1
		Requirements	None
	Name	Run_Obs1_ReadBlock_2	
	Requirements	None	
	Servercall points	**Name**	sscp_ReadBlock_2
		Type	SynchronousServerCallPoint
		Access to	Obs1_ReadBlock_2
		Requirements	None
	Name	Run_Obs1_ReadBlock_3	
	Requirements	None	
	Servercall points	**Name**	sscp_ReadBlock_3
		Type	SynchronousServerCallPoint
		Access to	Obs1_ReadBlock_3
		Requirements	None
Operation Invoked Event	**Name**	OperationInvokedEvent_Obs1_4	
	Runnable Entity	Run_Obs1_JobFinished_4	
		Access to	Obs1_JobFinished_4
		Requirements	None

SWC Name	Observation_SWC2		
PORTS	**Name**	Obs2_JobFinished_1	
	Type	PPortPrototype	
	Interface	JobFinished_1	
	Name	Obs2_JobFinished_2	
	Type	PPortPrototype	
	Interface	JobFinished_2	
Operation Invoked Event	**Name**	OperationInvokedEvent_Obs2	
	Runnable Entity	Run_Obs2_JobFinished_1	
		Access to	Obs2_JobFinished_1
		Requirements	None
	Name	OperationInvolkeEvent_Obs2_2	
	Runnable Entity	Run_Obs2_JobFinished_2	
		Access to	Obs2_JobFinished_2
		Requirements	None

Service Dependency

Control_SWC	**Name**	SwServiceDependency_Immediate
	RoleBasedPortAssignment	Ctrl_JobFinished_1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Ctrl_RestoreBlockDefaults_Client_1 (NvMService)
	RoleBasedPortAssignment	Ctrl_WriteBlock_1 (NvMService)
	RoleBasedPortAssignment	Ctrl_SetBlockProtection_1 (NvMAdmin)
	NvDataPort	PP_SendData3, PP_SendData1
	ServiceNeeds	NvBlockNeeds : storeImmediate – True nRomBlocks : 1
	Name	SwcServiceDependancy_Cyclic_Ctrl
	RoleBasedPortAssignment	Ctrl_GetErrorStatus_2 (NvMService)
	NvDataPort	PP_SendData2
	ServiceNeeds	NvBlockNeeds : storeImmediate : False

		storeCyclic : True cyclicWritingPeriod : 10 sec calcRamBlockCrc : True useCRCCompMechanism : True
	Name	SwcServiceDependancy_Cyclic_Shutdown_Ctrl
	NvDataPort	PP_SendData4, PP_SendData5
	ServiceNeeds	NvBlockNeeds : storeImmediate : False storeCyclic : True cyclicWritingPeriod : 10 sec storeAtShutDown : True useAutoValidationAtShutDown : True
	Name	SwcServiceDependancy_NoDirtyFlag_Ctrl
	RoleBasedPortAssignment	Ctrl_WriteBlock_4 (NvMService)
	NvDataPort	PP_SendData6
	ServiceNeeds	NvBlockNeeds : storeImmediate : False storeCyclic : False cyclicWritingPeriod : 10 sec storeAtShutDown : False useAutoValidationAtShutDown : False

Observation_SWC1	**Name**	SwServiceDependency_Immediate
	RoleBasedPortAssignment	Obs1_JobFinished_1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Obs1_ReadBlock_1 (NvMService)
	RoleBasedPortAssignment	Obs1_ReadBlock_3 (NvMService)
	NvDataPort	RP_ObserveData1, RP_ObserveData3
	Name	SwServiceDependency_Cyclic_Obs1
	RoleBasedPortAssignment	Obs1_JobFinished_2 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Obs1_ReadBlock_2 (NvMService)
	NvDataPort	RP_ObserveData2
	Name	SwServiceDependency_Cyclic_Shutdown_Obs1

	RoleBasedPortAssignment	Obs1_JobFinished_3 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Obs1_ReadBlock_3 (NvMService)
	NvDataPort	RP_ObserveData4, RP_ObserveData_5
	Name	SwServiceDependency_Cyclic_NoDirtyFlag_Obs1
	RoleBasedPortAssignment	Obs1_JobFinished_4 (NvMNotifyJobFinished)
	NvDataPort	RP_ObserveData8

Observation_SWC2	**Name**	SwServiceDependency_Immediate_Obs2
	RoleBasedPortAssignment	Obs2_JobFinished_1 (NvMNotifyJobFinished)
	NvDataPort	RP_ObserveData6
	Name	SwServiceDependency_Cyclic_Obs2
	RoleBasedPortAssignment	Obs2_JobFinished_2 (NvMNotifyJobFinished)
	NvDataPort	RP_ObserveData7

NvBlockSwComponentType	**Name**	NvBlockDescriptor1
	RoleBasedPortAssignment	Nv_JobFinished_1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_JobFinished_1_Obs1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_JobFinished_1_Obs2 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_SetBlockProtection_1 (NvMAdmin)
	RoleBasedPortAssignment	Nv_WriteBlock_1 (NvMService)
	RoleBasedPortAssignment	Nv_ReadBlock_1 (NvMService)
	NvRamBlockElement	PR_ReceiveData1, PR_ReceiveData3
	RamBlock	Nv_MainRecord {Nv_Data4; Nv_Record {Nv_Data1; Nv_Data2} }
	RomBlock	Nv_MainRecord {Nv_Data5; Nv_Record {Nv_Data1; Nv_Data2} }

		Initvalue : Nv_Data4 : 0x64 Nv_Record {Nv_Data1 : 0xAAAA Nv_Data2 : 0xBBBB }
	NvBlockNeeds	NvBlockDescriptor1Needs : storeImmediate – True nRomBlocks : 1
	Name	NvBlockDescriptor2
	RoleBasedPortAssignment	Nv_JobFinished_2_Obs1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_JobFinished_2_Obs2 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_ReadBlock_2 (NvMService)
	RoleBasedPortAssignment	Nv_GetErrorStatus_2 (NvMService)
	NvRamBlockElement	PR_ReceiveData2
	RamBlock	Nv_Array[4] : uint16
	NvBlockNeeds	NvBlockDescriptor2Needs : storeImmediate : False storeCyclic : True cyclicWritingPeriod : 10sec calcRamBlockCrc : True useCRCCompMechanism : True
	Name	NvBlockDescriptor3
	RoleBasedPortAssignment	Nv_JobFinished_3_Obs1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_ReadBlock_3 (NvMService)
	NvRamBlockElement	PR_ReceiveData4, PR_ReceiveData5
	RamBlock	Nv_Record1 { Nv_Data5 : uint16 ; Nv_Data3_Bitfield
	NvBlockNeeds	NvBlockDescriptor3Needs : storeImmediate : False storeCyclic : True storeAtShutdown : True cyclicWritingPeriod : 10sec
	Name	NvBlockDescriptor4
	RoleBasedPortAssignment	Nv_JobFinished_4_Obs1 (NvMNotifyJobFinished)
	RoleBasedPortAssignment	Nv_WriteBlock_4 (NvMService)
	NvRamBlockElement	PR_ReceiveData6

	RamBlock	Nv_Data6 : uint16
	NvBlockNeeds	NvBlockDescriptor4Needs : storeImmediate : False storeCyclic : False storeAtShutdown : False supportDirtyFlag : False

7.1.2.4 Mandatory vs Customizable Parts

Not Applicable

7.1.3 Test Case Design

Not Applicable

7.2 Re-usable Test Steps

None.

7.3 Test Cases

7.3.1 [ATS_RTE_01237] Implicit Write Access to NvDataElements - PrimitiveData Type of Write Immediate Block

Test Objective	Implicit Write Access to NvDataElements - PrimitiveData Type of Write Immediate Block		
ID	ATS_RTE_01237	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_03829 RTE: SWS_Rte_08083 RTE: SWS_Rte_08085 RTE: SWS_Rte_08086 RTE: SWS_Rte_08087 RTE: SWS_Rte_08088 RTE: SWS_Rte_08089		
Requirements / Reference to Test Environment	UC05.01, UC05.02		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationPrimitiveDataType App_Data5 : uint32		

	NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data4 : App_Data5 NvBlockSwComponentType has NvBlockDescriptor1: PR_ReceiveData1Event triggers NvM_WriteBlock. supportDirtyFlag : True storeImmediate : True Composition: Control_SWC.PP_SendData1 and NvBlockDescriptor1.PR_ReceiveData1 are connected. NvBlockDescriptor1.PR_ReceiveData1 and Observation_SWC1.RP_ObserveData1 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor1 NvBlockSwComponent Clientports : NvBlockDescriptor1 Server Ports : In Control_SWC, Observation_SWC1 C/S Interface for NvM_ReadBlock for NvBlockDescriptor1 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort
Summary	To check the NvDataHandling for Implicit Sender Receiver Communication for PrimitiveDataType with dirtyflag mechanism is enabled and storeImmediate is configured as True. Once Implicit Write is completed by Control_SWC, NvM_WriteBlock will be triggered for NvBlockDescriptor1 based on Data Receive Event. Wait until JobEndNotification for NvM_WriteBlock. Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor1. Once NvM_ReadBlock gets JobEndNotification, Observation_SWC1 reads its require port. Data read is the updated value in RAM Block by Control_SWC.
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor
Pre-conditions	None
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_1 starts
Step 2	**[RUN<Run_Ctrl_1>]** Run_Ctrl_1 executes Rte_IWrite on VDP_Nv1 with value as below: App_Data5 = 0x0A
Step 3	**[CP]** Run_Ctrl_1 is preempted by Run_Obs1_1
Step 4	**[RUN<Run_Obs1_1>]**	**[RUN<Run_Obs1_1>]**

	Run_Obs1_1 executes Rte_IRead from VDP_Nv1	Read data is initial value of NvBlock element and not the updated value from Control_SWC. Nv_Data4 = 0x190
Step 5	**[CP]** Run_Obs1_1 terminates Run_Ctrl_1 is resumed.	
Step 6	**[CP]** Run_Ctrl_1 terminates	
Step 7	**[SWC <NvBlockSwComponent>]** PR_ReceiveData1Event invokes Run_NvM_WriteBlock which triggers NvM_WriteBlock for ImmediateBlock.	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK.
Step 8	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_1 has been called for NvMBlockDescriptor1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 9	**[CP]** Run_Obs1_ReadBlock_1 starts	
Step 10	**[RUN<Run_Obs1_ReadBlock_1>** Run_Obs1_ReadBlock_1 executes Rte_Call of NvM_ReadBlock() for NvMBlockDescriptor1	**[RUN<Run_Obs1_ReadBlock_1]** NvM_ReadBlock() should return E_OK.
Step 11	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_1 has been called for NvMBlockDescriptor1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 12	**[CP]** Run_Obs1_1 starts	
Step 13	**[RUN<Run_Obs1_1>]** Run_Obs1_1 executes Rte_IRead from VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is as below: Nv_Data4 = 0x0A
Post-conditions	None	

7.3.2 [ATS_RTE_01252] Explicit Write Access to NvDataElements - Composite type (Array) of Write Cyclic Block

Test Objective	Explicit Write Access to NvDataElements - Composite type (Array) of Write Cyclic Block		
ID	ATS_RTE_01252	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_03829 RTE: SWS_Rte_03833 RTE: SWS_Rte_08082 RTE: SWS_Rte_08084 RTE: SWS_Rte_08086 RTE: SWS_Rte_08087 RTE: SWS_Rte_08088 RTE: SWS_Rte_08089		
Requirements / Reference to Test Environment	UC05.01, UC05.02		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationArrayDataType App_Array[4] : uint16 NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Array[4] : App_Array[4] NvBlockSwComponentType has NvBlockDescriptor2 which is configured as storeCyclic NvM_WriteBlock will be triggered periodically. supportDirtyFlag : True Composition: Control_SWC.PP_SendData2 and NvBlockDescriptor2.PR_ReceiveData2 are connected. NvBlockDescriptor2.PR_ReceiveData2 and Observation_SWC1.RP_ObserveData2 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor2 NvBlockSwComponent : Clientport for NvBlockDescriptor2 Observation_SWC1 and Observation_SWC2 - Server Ports C/S Interface for NvM_ReadBlock for NvBlockDescriptor2 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort		

Summary	To check the NvDataHandling for Explicit Sender Receiver Communication for Composite Type like Arrays with dirtyflag mechanism is enabled and storeCyclic is configured as True. Once Explicit Write is completed by Control_SWC, NvM_WriteBlock will be triggered for NvBlockDescriptor2 in the next periodic activity based on Timing Event. Wait until JobEndNotification for NvM_WriteBlock. Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor2. Once NvM_ReadBlock gets JobEndNotification, Observation_SWC1 reads its require port. Data read is the updated value in RAM Block by Control_SWC.	
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_2 starts	
Step 2	**[RUN<Run_Ctrl_2>]** Run_Ctrl_2 executes Rte_Write on VDP_Nv2 with value as below: App_Array[0] = 0xAABB App_Array[1] = 0xBBCC App_Array[2] = 0xCCDD App_Array[3] = 0xDDEE	**[RUN<Run_Ctrl_2>]** Rte_Write should return RTE_E_OK.
Step 3	**[CP]** Run_Ctrl_2 is preempted by Run_Obs1_2	
Step 4	**[RUN<Run_Obs1_2>]** Run_Obs1_2 executes Rte_Read from VDP_Nv2	**[RUN<Run_Obs1_2>]** Read data is updated value from Control_SWC Nv_Array[0] = 0xAABB Nv_Array[1] = 0xBBCC Nv_Array[2] = 0xCCDD Nv_Array[3] = 0xDDEE
Step 5	**[CP]**	

	Run_Obs1_2 terminates. Run_Ctrl_2 is resumed.	
Step 6	**[CP]** Run_Ctrl_2 terminates	
Step 7	**[SWC <NvBlockSwComponent>]** Timing Event PR_TimingData2Event invokes Run_NvM_WriteBlock_2 in the next periodic activity which triggers NvM_WriteBlock	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK
Step 8	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvMBlockDescriptor2 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 9	**[CP]** Run_Obs1_ReadBlock_2 starts	
Step 10	**[RUN<Run_Obs1_ReadBlock_2>]** Run_Obs1_ReadBlock_2 executes Rte_Call of NvM_ReadBlock() for NvMBlockDescriptor2	**[RUN<Run_Obs1_ReadBlock_2>]** NvM_ReadBlock() should return E_OK
Step 11	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvMBlockDescriptor2 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 12	**[CP]** Run_Obs1_2 starts	
Step 13	**[RUN<Run_Obs1_2>]** Run_Obs1_2 executes Rte_Read from VDP_Nv2	**[RUN<Run_Obs1_2>]** Read data is as below: Nv_Array[0] = 0xAABB Nv_Array[1] = 0xBBCC Nv_Array[2] = 0xCCDD Nv_Array[3] = 0xDDEE
Post-conditions	None	

7.3.3 [ATS_RTE_01283] Implicit Write Access to NvDataElements - Structure Data Type of Write Immediate Block

Test Objective	Implicit Write Access to NvDataElements - Structure Data Type of Write Immediate Block		
ID	ATS_RTE_01283	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	Rte, NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_03829 RTE: SWS_Rte_03830 RTE: SWS_Rte_08082 RTE: SWS_Rte_08087		
Requirements / Reference to Test Environment	UC05.01, UC05.02		

Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1
	Control_SWC :
	ApplicationRecordDataType
	App_Record : Record
	Record
	{
	App_Data1 : uint16
	App_Data2 : uint16
	App_Data3 : uint16
	App_Data4 : uint16
	}
	
	NvBlockSwComponentType and Observation_SWC1 has following interface:
	Nv_Record
	{
	Nv_Data1 : uint16
	Nv_Data2 : uint16
	}
	NvBlockSwComponentType has NvBlockDescriptor1:
	PR_ReceiveData3Event triggers NvM_WriteBlock.
	supportDirtyFlag : True
	storeImmediate : True
	
	Composition:Control_SWC.PP_SendData3 and NvBlockDescriptor1.PR_ReceiveData3 are connected.
	NvBlockDescriptor1.PR_ReceiveData3 and Observation_SWC1.RP_ObserveData3 are connected.
	
	PortInterfaceMapping :
	App_Data1 -> Nv_Data1
	App_Data2 -> Nv_Data2
	
	NvBlockDescriptor1
	C/S Interface for JobFinished
	NvBlockSwComponent : Clientport
	Observation_SWC1 - Server Ports		

	C/S Interface for NvM_ReadBlock NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort
Summary	To check the NvDataHandling for Implicit Sender Receiver Communication for Composite types like C Structures with dirtyflag mechanism is enabled and storeImmediate is configured as True. Once Implicit Write is completed by Control_SWC, NvM_WriteBlock will be triggered for NvBlockDescriptor1 based on Data Receive Event. Wait until JobEndNotification for NvM_WriteBlock. Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor1. Once NvM_ReadBlock gets JobEndNotification, Observation_SWC1 reads its require port. Data read is the updated value in RAM Block by Control_SWC.
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_3 starts	

Step 2	**[RUN<Run_Ctrl_3>]**
	
	Run_Ctrl_3 executes Rte_IWrite on VDP_ImplWrite2 with value as below :
	
	{
	
	App_Data1 = 0x55AA
	
	App_Data2 = 0xAA55
	
	App_Data3 = 0x5A5A
	
	App_Data4 = 0xA5A5
	
	}	

Step 3	**[CP]** Run_Ctrl_3 terminates	
Step 4	**[SWC <NvBlockSwComponent>]** PR_ReceiveData3Event invokes Run_NvM_WriteBlock which triggers NvM_WriteBlock for immediate block.	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK
Step 5	**[CP]**	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of

	Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	Run_Obs1_JobFinished_1 has been called for NvMBlockDescriptor1 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 6	**[CP]** Run_Obs1_ReadBlock_1 starts	
Step 7	**[RUN<Run_Obs1_ReadBlock_1>]** Run_Obs1_ReadBlock_1 executes Rte_Call of NvM_ReadBlock() for NvBlockDescriptor1	**[RUN<Run_Obs1_ReadBlock_1>]** NvM_ReadBlock() should return E_OK.
Step 8	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_1 has been called for NvMBlockDescriptor1 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 9	**[CP]** Run_Obs1_3 starts	
Step 10	**[RUN<Run_Obs1_3>]** Run_Obs1_3 executes Rte_IRead from VDP_Nv3	**[RUN<Run_Obs1_3>]** Read data is as below: Nv_Data1 = 0x55AA Nv_Data2 = 0xAA55
Post-conditions	None	

7.3.4 [ATS_RTE_01253] Explicit Write Access to NvDataElements - Boolean to BitField Type of Write Cyclic Block

Test Objective	Explicit Write Access to NvDataElements - Boolean to BitField Type of Write Cyclic Block		
ID	ATS_RTE_01253	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_03831 RTE: SWS_Rte_08082 RTE: SWS_Rte_08086 RTE: SWS_Rte_08087		
Requirements / Reference	UC05.01, UC05.02		

to Test Environment	
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationPrimitiveDataType App_boolean_element : boolean NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data3_bitfield : uint8 NvBlockSwComponentType has NvBlockDescriptor3 which is configured as storeCyclic and storeAtShutdown NvM_WriteBlock will be triggered periodically supportDirtyFlag : True Composition: Control_SWC.PP_SendData4 and NvBlockDescriptor3.PR_ReceiveData4 are connected. NvBlockDescriptor3.PR_ReceiveData4 and Observation_SWC1.RP_ObserveData4 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor3 NvBlockSwComponent : Clientport for NvBlockDescriptor3 Observation_SWC1 - Server Ports C/S Interface for NvM_ReadBlock for NvBlockDescriptor3 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort
Summary	To check the NvDataHandling for Explicit Sender Receiver Communication for Boolean to Bitfield data type with dirtyflag mechanism is enabled. Once Explicit Write is completed by Control_SWC, NvM_WriteBlock will be triggered for NvBlockDescriptor3 in the next periodic activity based on Timing Event since service dependency for this data type is storeCyclic. Wait until JobEndNotification for NvM_WriteBlock. Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor3. Once NvM_ReadBlock gets JobEndNotification, Observation_SWC1 reads its require port. Data read is the updated value in RAM Block in respective bitfiled by Control_SWC.
Needed Adaptation to other Releases	NA
Pre-conditions	None

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_4 starts	
Step 2	**[RUN<Run_Ctrl_4>]**	**[RUN<Run_Ctrl_4>]** Rte_Write should return RTE_E_OK.

	Run_Ctrl_4 executes Rte_Write on VDP_ExplWrite2 with value as below: App_Boolean_Data = True	
Step 3	**[CP]** Run_Ctrl_4 terminates	
Step 4	**[SWC<NvBlockSwComponent>]** PR_TimingData4Event invokes Run_NvM_WriteBlock_3 in the next periodic activity which triggers NvM_SetRamBlockStatus for NvBlockDescriptor3 with blockchanged parameter as True	**[SWC<NvBlockSwComponent>]** NvM_SetRamBlockStatus should return E_OK
Step 5	**[SWC <NvBlockSwComponent>]** Run_NvM_WriteBlock_3 triggers NvM_WriteBlock	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK.
Step 6	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_3 has been called for NvBlockDescriptor3 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 7	**[CP]** Run_Obs1_ReadBlock_3 starts	
Step 8	**[RUN<Run_Obs1_ReadBlock_3>]** Run_Obs1_ReadBlock_3 executes Rte_Call of NvM_ReadBlock() for NvBlockDescriptor3	**[RUN<Run_Obs1_ReadBlock_3>]** NvM_ReadBlock() should return E_OK.
Step 9	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvBlockDescriptor3 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 10	**[CP]** Run_Obs1_4 starts	
Step 11	**[RUN<Run_Obs1_4>]** Run_Obs1_4 executes Rte_Read from VDP_Nv4	**[RUN<Run_Obs1_4>]** Read data is as below: Nv_Data3_Bitfield = 0x08 Hint : Computation Method - Sets the 3rd bit in RAM Block.
Post-conditions	None	

7.3.5 [ATS_RTE_01254] Explicit Write Access to NvDataElements - Primitive Data Type of Write At Shutdown Block

Test Objective	Explicit Write Access to NvDataElements - Primitive Data Type of Write At Shutdown Block		
ID	ATS_RTE_01254	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00453 RTE: SWS_Rte_03831 RTE: SWS_Rte_08080 RTE: SWS_Rte_08088		
Requirements / Reference to Test Environment	UC05.01, UC05.02, UC05.03		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationPrimitiveDataType App_Data6 : uint16 NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data5 : App_Data6 NvBlockSwComponentType has NvBlockDescriptor3 which is configured as below: storeCyclic : True storeAtShutdown : True useAutoValidationAtShutDown : True supportDirtyFlag : True Composition: Control_SWC.PP_SendData5, NvBlockDescriptor3.PR_ReceiveData5 and Observation_SWC1.RP_ObserveData5 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor3 NvBlockSwComponent : Clientport for NvBlockDescriptor3 Observation_SWC1 - Server Ports C/S Interface for NvM_ReadBlock for NvBlockDescriptor3 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort		
Summary	To check the NvDataHandling for Explicit Sender Receiver Communication for Primitive Data Type with dirtyflag mechanism is enabled. Once Explicit Write is completed by Control_SWC, NvM_SetRamBlockStatus will be invoked by Data Received Event since service dependency for this data type is storeAtShutdown. Power off Ecu and Power on the same.		

	Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor3. Once NvM_ReadBlock gets JobEndNotification, Observation_SWC1 reads its require port. Data read is the updated value in RAM Block by Control_SWC.	
Needed Adaptation to other Releases	NA	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_5 starts	
Step 2	**[RUN<Run_Ctrl_5>]** Run_Ctrl_5 executes Rte_Write on VDP_Nv5 with value as below : App_Data6 = 0x5A5A	**[RUN<Run_Ctrl_5>]** Rte_Write should return RTE_E_OK.
Step 3	**[CP]** Run_Ctrl_5 terminates	
Step 4	**[SWC <NvBlockSwComponent>]** PR_TimingData5Event invokes Run_WriteBlock_5 which triggers NvM_SetRamBlockStatus with blockchanged parameter as True	**[SWC <NvBlockSwComponent>]** NvM_SetRamBlockStatus should return E_OK
Step 5	**[SWC<NvBlockSwComponent>]** Run_WriteBlock_5 triggers NvM_WriteBlock	**[SWC<NvBlockSwComponent>]** NvM_WriteBlock should return E_OK
Step 6	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_3 has been called for NvBlockDescriptor3 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 7	**[CP]** NvM_WriteAll shall be triggered by BswM during ECU shutdown	**[SWC<NvBlockSwComponent>]** MultiBlockJobStatusInformation (NvM_WriteAll) has been called with ServiceId value 0x0d and job_result NVM_REQ_OK
Step 8	**[LT<PowerSupply>]** Power off ECU	
Step 9	**[CP]** Wait for n seconds [n depends on time needed for platform to shutdown]	
Step 10	**[LT<PowerSupply>]** Power on ECU	

Step 11	**[CP]** Run_Obs1_ReadBlock_3 starts	
Step 12	**[RUN<Run_Obs1_ReadBlock_3>]** Run_Obs1_ReadBlock_3 executes Rte_Call of NvM_ReadBlock() for NvBlockDescriptor3	**[RUN<Run_Obs1_ReadBlock_3>]** NvM_ReadBlock() should return E_OK
Step 13	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_3 has been called for NvBlockDescriptor3 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 14	**[CP]** Run_Obs1_5 starts	
Step 15	**[RUN<Run_Obs1_5>]** Run_Obs1_5 executes Rte_Read from VDP_Nv5	**[RUN<Run_Obs1_5>]** Read data is as below: Nv_Data5 = 0x5A5A
Post-conditions	None	

7.3.6 [ATS_RTE_01255] Write Optimization if useCRCCompMechanism is True

Test Objective	Write Optimization if useCRCCompMechanism is True		
ID	ATS_RTE_01255	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	RTE: SWS_Rte_03829 RTE: SWS_Rte_08082 RTE: SWS_Rte_08086 RTE: SWS_Rte_08087		
Requirements / Reference to Test Environment	UC05.01, UC05.02, UC05.03		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationArrayDataType App_Array[4] : uint16 NvBlockSwComponentType and Observation_SWC1 has following interface:		

	Nv_Array[4] : App_Array[4] NvBlockSwComponentType has NvBlockDescriptor2 which is configured as storeCyclic NvM_WriteBlock will be triggered periodically. supportDirtyFlag : True Composition: Control_SWC.PP_SendData2, NvBlockDescriptor2.PR_ReceiveData2 and Observation_SWC1.RP_ObserveData2 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor2 NvBlockSwComponent : Clientport for NvBlockDescriptor2 Observation_SWC1 - Server Ports C/S Interface for NvM_ReadBlock for NvBlockDescriptor2 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort C/S Interface for NvM_GetErrorStatus for NvBlockDescriptor2 NvBlockSwComponent : ServerPort Control_SWC : ClientPort	
Summary	To check the CRCCompMechanism feature in NvDataHandling for Composite Type like Arrays with dirtyflag mechanism is enabled. Once Explicit Write is completed by Control_SWC, NvM_WriteBlock will be triggered for NvBlockDescriptor2 in the next periodic activity based on Timing Event. Invoke NvM_WriteAll without changing the data in RAM Block. Wait until 5 seconds. Control_SWC should invoke NvM_GetErrorStatus for NvBlockDescriptor2. JobResult for the block should return NVM_REQ_BLOCK_SKIPPED because CRC matches with last Write Job.	
Needed Adaptation to other Releases	NA	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_2 starts	
Step 2	**[RUN<Run_Ctrl_2>]** Run_Ctrl_2 executes Rte_Write on VDP_Nv2 with value as below: App_Array[0] = 0x1A1A App_Array[1] = 0x2B2B App_Array[2] = 0x3C3C App_Array[3] = 0x4D4D	**[RUN<Run_Ctrl_2>]** Rte_Write should return RTE_E_OK

Step 3	**[CP]** Run_Ctrl_2 terminates	
Step 4	**[SWC <NvBlockSwComponent>]** PR_TimingData2Event invokes Run_NvM_WriteBlock_2 in the next periodic activity which triggers NvM_WriteBlock.	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK.
Step 5	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvBlockDescriptor2 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 6	**[CP]** Run_Obs1_ReadBlock_2 starts	
Step 7	**[RUN<Run_Obs1_ReadBlock_2>]** Run_Obs1_ReadBlock_2 executes Rte_Call of NvM_ReadBlock() for NvBlockDescriptor2	**[RUN<Run_Obs1_ReadBlock_2>]** NvM_ReadBlock() should return E_OK
Step 8	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvBlockDescriptor2 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 9	**[CP]** Run_Obs1_2 starts	
Step 10	**[RUN<Run_Obs1_2>]** Run_Obs1_2 executes Rte_Read from VDP_Nv2	**[RUN<Run_Obs1_2>]** Read data is as below: Nv_Array[0] = 0x1A1A Nv_Array[1] = 0x2B2B Nv_Array[2] = 0x3C3C Nv_Array[3] = 0x4D4D
Step 11	**[CP]** Invoke NvM_WriteAll	
Step 12	**[CP]** Wait for 5seconds.	
Step 13	**[CP]** Run_Ctrl_GetError_2 starts	

Step 14	**[RUN<Run_Ctrl_GetError_2>]** Run_Ctrl_GetError_2 executes Rte_Call of NvM_GetErrorStatus for NvBlockDescriptor2	**[SWC<Control_SWC>]** 1. NvM_GetErrorStatus should return E_OK. 2. RequestResultPtr parameter should be updated with NVM_REQ_BLOCK_SKIPPED for NvBlockDescriptor2.
Post-conditions	None	

7.3.7 [ATS_RTE_01284] Implicit Read Access of Ram Block from Multiple SWC's

Test Objective	Implicit Read Access of Ram Block from Multiple SWC's		
ID	ATS_RTE_01284	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	Rte, NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_03829 RTE: SWS_Rte_08085 RTE: SWS_Rte_08086		
Requirements / Reference to Test Environment	UC05.01, UC05.02		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType, Observation_SWC1 and Observation_SWC2 Control_SWC with Sender Interface: Primitive Data Type App_Data5 : uint32 NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data4 : uint32 NvBlockSwComponentType has NvBlockDescriptor1: PR_ReceiveData1Event triggers NvM_WriteBlock. Composition: Control_SWC.PP_SendData1 and NvBlockDescriptor1.PR_ReceiveData1 are connected. NvBlockDescriptor1.PR_ReceiveData1 and Observation_SWC1.RP_ObserveData1 are connected. NvBlockDescriptor1.PR_ReceiveData1 and Observation_SWC2.RP_ObserveData6 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor1 NvBlockSwComponent : Clientport for NvBlockDescriptor1 Control_SWC, Observation_SWC1 and Observation_SWC2 - Server Ports		

Summary	To check the NvDataHandling for Implicit Read from Multiple SWC's for Primitive Data Type if dirtyflag mechanism is enabled. Control Starts, does an implicit write on its provide port. NvM_WriteBlock will be invoked by the Data Receive Event once implicit write is completed. Wait until JobFinished for NvM_WriteBlock. Observation_SWC1 reads its require port. Data read is the actual value written from Control SWC. Observation_SWC2 reads its require port. Data read is the actual value written from Control SWC.	
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_1 starts	
Step 2	**[RUN<Run_Ctrl_1>]** Run_Ctrl_1 executes Rte_IWrite on VDP_Nv1 with value as below: App_Data5 = 0x05	
Step 3	**[CP]** Run_Ctrl_1 is preempted by Run_Obs1_1	
Step 4	**[RUN<Run_Obs1_1>]** Run_Obs1_1 executes Rte_IRead on VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is initial value in RAMBlock and not the updated value from Control_SWC. Nv_Data4 = 0x190
Step 5	**[CP]** Run_Obs1_1 terminates Run_Ctrl_1 is resumed.	
Step 6	**[CP]** Run_Ctrl_1 is preempted by Run_Obs2_1	
Step 7	**[RUN<Run_Obs2_1>]** Run_Obs2_1 executes Rte_IRead on VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is initial value in RAMBlock and not the updated value from Control_SWC. Nv_Data4 = 0x190
Step 8	**[CP]**	

	Run_Obs2_1 terminates. Run_Ctrl_1 is resumed.	
Step 9	**[CP]** Run_Ctrl_1 terminates	
Step 10	**[SWC <NvBlockSwComponent>]** PR_ReceiveData1Event triggers Run_NvM_WriteBlock which triggers NvM_WriteBlock for immediate block.	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK.
Step 11	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_1 has been called for NvBlockDescriptor1 with serviceid value 0x7 and job_result NVM_REQ_OK.
Step 12	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC2>]** NvMNotifyJobFinished server operation of Run_Obs2_JobFinished_1 has been called for NvBlockDescriptor1 with serviceid value 0x7 and job_result NVM_REQ_OK.
Step 13	**[CP]** Run_Obs1_1 starts	
Step 14	**[RUN<Run_Obs1_1>]** Run_Obs1_1 executes Rte_IRead on VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is as below: Nv_Data4 = 0x05
Step 15	**[CP]** Run_Obs2_1 starts	
Step 16	**[RUN<Run_Obs2_1>]** Run_Obs2_1 executes Rte_IRead on VDP_Nv1	**[RUN<Run_Obs2_1>]** Read data is as below: Nv_Data4 = 0x05
Post-conditions	None	

7.3.8 [ATS_RTE_01285] Explicit Read Access of Nv Data Elements from Multiple SWC's

Test Objective	Explicit Read Access of Nv Data Elements from Multiple SWC's		
ID	ATS_RTE_01285	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2

Affected Modules	Rte, NvM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129 ATR: ATR_ATR_00130		
Trace to SWS Item	RTE: SWS_Rte_03830 RTE: SWS_Rte_03833 RTE: SWS_Rte_08082		
Requirements / Reference to Test Environment	UC05.01, UC05.02		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType, Observation_SWC1 and Observation_SWC2 Control_SWC with Sender Interface: Array VDP_Nv2 : App_Array[4] (ApplicationArrayDataType) NvBlockSwComponentType, Observation_SWC1 and Observation_SWC2 has following interface: Nv_Array[4] : App_Array[4] NvBlockSwComponentType has NvBlockDescriptor2 which is configured as storeCyclic NvM_WriteBlock will be triggered periodically. supportDirtyFlag : True Composition: Control_SWC.PP_SendData2 and NvBlockDescriptor2.PR_ReceiveData2 are connected. NvBlockDescriptor2.PR_ReceiveData2 and Observation_SWC1.RP_ObserveData2 are connected. NvBlockDescriptor2.PR_ReceiveData2 and Observation_SWC2.RP_ObserveData7 are connected. C/S Interface for JobFinished from NvM for NvBlockDescriptor2 NvBlockSwComponent : Clientport for NvBlockDescriptor2 Observation_SWC1 and Observation_SWC2 - Server Ports		
Summary	To check the NvDataHandling for Explicit Read from Multiple SWC's for Composite Type like Arrays if dirtyflag mechanism is enabled. Control Starts, does an Explicit Write on the Provided Port. NvM_WriteBlock will be invoked by the Timing Event in the next periodic activity. Wait until JobFinished for NvM_WriteBlock. Observation_SWC1 reads its require port. Data read is the actual value written from Control SWC. Observation terminates. Observation_SWC2 reads its require port. Data read is the actual value written from Control SWC. Observation terminates.		
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor		
Pre-conditions	None		
Main Test Execution			

Test Steps		Pass Criteria
Step 1	**[CP]** Run_Ctrl_2 starts	
Step 2	**[RUN<Run_Ctrl_2>]** Run_Ctrl_2 executes Rte_Write on VDP_Nv2 with value as below: App_Array[0] = 0x5A5A App_Array[1] = 0xA5A5 App_Array[2] = 0x55AA App_Array[3] = 0xAA55	
Step 3	**[CP]** Run_Ctrl_2 is preempted by Run_Obs1_2	
Step 4	**[RUN<Run_Obs1_2>]** Run_Obs1_2 executes Rte_Read from VDP_Nv2	**[RUN<Run_Obs1_2>]** Read data is updated value from Control_SWC Nv_Array[0] = 0x5A5A Nv_Array[1] = 0xA5A5 Nv_Array[2] = 0x55AA Nv_Array[3] = 0xAA55
Step 5	**[CP]** Run_Obs1_2 terminates. Run_Ctrl_2 is resumed.	
Step 6	**[CP]** Run_Ctrl_2 terminates	
Step 7	**[SWC <NvBlockSwComponent>]** PR_TimingData2Event invokes Run_NvM_WriteBlock_2 in the next periodic activity which triggers NvM_WriteBlock.	**[SWC <NvBlockSwComponent>]** NvM_WriteBlock should return E_OK.
Step 8	**[CP]** Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_2 has been called for NvBlockDescriptor2 with serviceid value 0x7 and job_result NVM_REQ_OK.
Step 9	**[CP]**	**[SWC<Observation_SWC2>]**

	Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	NvMNotifyJobFinished server operation of Run_Obs2_JobFinished_2 has been called for NvBlockDescriptor2 with serviceid value 0x7 and job_result NVM_REQ_OK.
Step 10	**[CP]** Run_Obs1_2 starts	
Step 11	**[RUN<Run_Obs1_2>]** Run_Obs1_2 executes Rte_Read on VDP_Nv2	**[RUN<Run_Obs1_2>]** Read data is as below: Nv_Array[0] = 0x5A5A Nv_Array[1] = 0xA5A5 Nv_Array[2] = 0x55AA Nv_Array[3] = 0xAA55
Step 12	**[CP]** Run_Obs2_2 starts	
Step 13	**[RUN<Run_Obs2_2>]** Run_Obs2_2 executes Rte_Read on VDP_Nv2	**[RUN<Run_Obs2_2>]** Read data is as below: Nv_Array[0] = 0x5A5A Nv_Array[1] = 0xA5A5 Nv_Array[2] = 0x55AA Nv_Array[0] = 0xAA55
Post-conditions	None	

7.3.9 [ATS_RTE_01261] NvMNotification and InitNotifyCallback using ClientServer Interface

Test Objective	NvMNotification and InitNotifyCallback using ClientServer Interface		
ID	ATS_RTE_01261	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance	ATR: ATR_ATR_00129		

Test Document		
Trace to SWS Item	NVRAMManager: SWS_NvM_00456 RTE: SWS_Rte_07625 RTE: SWS_Rte_07626 RTE: SWS_Rte_07629 RTE: SWS_Rte_07630 RTE: SWS_Rte_07631	
Requirements / Reference to Test Environment	UC05.01	
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC Client Port : Ctrl_RestoreBlockDefaults_Client_1 Server Port : Ctrl_JobFinished_1 NvBlockSwComponent Server Port : Nv_RestoreBlockDefaults_1 Client Port : Nv_JobFinished_1 NvBlockDescriptor1 nROMBlocks : 1 should be configured with default values for respective data types Vdp_Nv1 and VDP_ImplWrite3 Observation_SWC1 Server Port : Obs1_JobFinished_1 NvMNotifyJobFinished for NvBlockDescriptor1 is mapped to Control_SWC and Observation_SWC1	
Summary	This testcase is to test the NvMJobFinished and InitNotifyCallback using NvM_RestoreBlockDefaults api of NvM. Control_SWC should invokes NvM_RestoreBlockDefaults api. NvM will provide Job End Notification to Rte and Rte should invoke NvMNotifyJobFinished with RequestResult to Control_SWC and Observation_SWC1. Observation starts by Observation_SWC1 and reads its require port. Data read is the default value of respective RAMBlock.	
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_RestoreBlockDefaults_1 starts	

Step 2	**[RUN<Run_Ctrl_RestoreBlockDefaults_1>]** Run_Ctrl_RestoreBlockDefaults_1 executes Rte_Call of NvM_RestoreBlockDefaults() for NvBlockDescriptor1	**[RUN<Run_Ctrl_RestoreBlockDefaults_1>]** NvM_RestoreBlockDefaults() should return E_OK
Step 3	**[CP]** Wait until InitBlockCallback from NvM	
Step 4	**[SWC <NvBlockSwComponent>]** Copy the default value from ROMBlock in NvMBlockDescriptor1 to RAMBlock	
Step 5	**[CP]** Wait until JobEndNotification (NvM_RestoreBlockDefaults) from NvM Check call for NvMNotifyJobFinished	**[SWC<Control_SWC>]** NvMNotifyJobFinished server operation of Run_Ctrl_JobFinished_1 has been called for NvBlockDescriptor1 with ServiceId value 0x8 and job_result NVM_REQ_OK.
Step 6	**[CP]** Wait until JobEndNotification (NvM_RestoreBlockDefaults) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_1 has been called for NvBlockDescriptor1 with ServiceId value 0x8 and job_result NVM_REQ_OK.
Step 7	**[CP]** Run_Obs1_1 starts	
Step 8	**[RUN<Run_Obs1_1>]** Run_Obs1_1 executes Rte_IRead from VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is a default value as below: Nv_Data4 = 0x64
Step 9	**[CP]** Run_Obs1_3 starts	
Step 10	**[RUN<Run_Obs1_3>]** Run_Obs1_3 executes Rte_IRead from VDP_Nv3	**[RUN<Run_Obs1_3>]** Read data for Record is default value as below: Nv_Data1 = 0xAAAA Nv_Data2 = 0xBBBB
Post-conditions	None	

7.3.10 [ATS_RTE_01262] Initialization of RAMBlock with Initivalue of VDP

Test Objective	Initialization of RAMBlock with Initivalue of VDP

ID	ATS_RTE_01262	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	Rte	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129		
Trace to SWS Item	RTE: SWS_Rte_03852 RTE: SWS_Rte_07046 RTE: SWS_Rte_07632		
Requirements / Reference to Test Environment	UC05.01		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationPrimitiveDataType App_Data5 : uint32 NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data4 : App_Data5 NvBlockSwComponentType has NvBlockDescriptor1: Composition: Control_SWC.PP_SendData1 and NvBlockDescriptor1.PR_ReceiveData1 are connected. NvBlockDescriptor1.PR_ReceiveData1 and Observation_SWC1.RP_ObserveData1 are connected.		
Summary	This testcase is to test the initialization of RAMBlock with initvalue of corresponding Variable Data Port while RTE starts. At the time of RTE Initialization, RAMBlock configured in the NvMBlockDescriptor will be initialized with initvalue configured for VDP. SWC should invoke Implicit Read to check the corresponding values are initialized properly.		
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor		
Pre-conditions	Rte_Init should be executed which initializes the VDP_Nv1 VDP_Nv1 = 0x190		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[CP]** Run_Obs1_1 starts		
Step 2	**[RUN<Run_Obs1_1>]** Run_Obs1_1 executes Rte_IRead from VDP_Nv1	**[RUN<Run_Obs1_1>]** Read data is as below: Nv_Data4 = 0x190	
Post-conditions	None		

7.3.11 [ATS_RTE_01263] Rejection of NvM_WriteBlock by NvM if BlockProtection is enabled by SWC

Test Objective	Rejection of NvM_WriteBlock by NvM if BlockProtection is enabled by SWC		
ID	ATS_RTE_01263	**AUTOSAR Releases**	4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129		
Trace to SWS Item	NVRAMManager: SWS_NvM_00737		
Requirements / Reference to Test Environment	UC05.01		
Configuration Parameters	This testcase uses : Control_SWC and NvBlockSwComponentType Control_SWC Client Port : Ctrl_WriteBlock_1 Client Port : Ctrl_SetBlockProtection_1 Server Port : Ctrl_JobFinished_1 NvBlockSwComponent Server Port : Nv_WriteBlock_1 Server Port : Nv_SetBlockProtection_1 Client Port : Nv_JobFinished_1		
Summary	This testcase is to test the Rejection of NvM_WriteBlock call fromControl_SWC by NvM when respective block is protected. Control_SWC invokes NvM_SetBlockProtection with ProtectionEnabled as True. Control_SWC invokes NvM_WriteBlock for protected block. NvM should return E_NOT_OK and report DEM Error NVM_E_WRITE_PROTECTED.		
Needed Adaptation to other Releases	NA		
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_SetBlockProtection_1 starts		
Step 2	**[RUN<Run_Ctrl_SetBlockProtection_1>]**		**[RUN<Run_Ctrl_SetBlockProtection_1>]**

	Run_Ctrl_SetBlockProtection_1 executes Rte_Call of NvM_SetBlockProtection() for NvBlockDescriptor1 with protectionenabled parameter as True	NvM_SetBlockProtection should return E_OK.
Step 3	**[CP]** Run_Ctrl_WriteBlock_1 starts	
Step 4	**[RUN<Run_Ctrl_WriteBlock_1>]** Run_Ctrl_WriteBlock_1 executes Rte_Call of NvM_WriteBlock() for NvBlockDescriptor1 which is write protected	**[RUN<Run_Ctrl_WriteBlock_1>]** NvM_WriteBlock should return E_NOT_OK.
Step 5	**[SWC<Control_SWC>]** Check status of DEM Event NVM_E_WRITE_PROTECTED.	**[SWC<Control_SWC>]** Status Bit "TestFailed" of DEM Event NVM_E_WRITE_PROTECTED is set.
Post-conditions	None	

7.3.12 [ATS_RTE_01288] Write Access to NvDataElements if storeDirtyFlag is False

Test Objective	Write Access to NvDataElements if storeDirtyFlag is False		
ID	ATS_RTE_01288	**AUTOSAR Releases**	4.1.1|4.2.1|4.2.2
Affected Modules	RTE, NVM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00129		
Trace to SWS Item	NVRAMManager: SWS_NvM_00455 RTE: SWS_Rte_07668		
Requirements / Reference to Test Environment	UC05.01, UC05.02		
Configuration Parameters	This testcase uses : Control_SWC, NvBlockSwComponentType and Observation_SWC1 Control_SWC with Sender Interface: ApplicationPrimitive Data Type App_Data7 : uint16 NvBlockSwComponentType and Observation_SWC1 has following interface: Nv_Data6 : App_Data7 NvBlockSwComponentType has NvBlockDescriptor4 which is configured as below: supportDirtyFlag : False storeCyclic : False storeAtShutdown : False Composition: Control_SWC.PP_SendData6,		

	NvBlockDescriptor4.PR_ReceiveData6 and Observation_SWC1.RP_ObserveData8 are connected. C/S Interface for NvM_WriteBlock for NvBlockDescriptor4 NvBlockSwComponent : ServerPort Control_SWC : ClientPort C/S Interface for JobFinished from NvM for NvBlockDescriptor4 NvBlockSwComponent : Clientport for NvBlockDescriptor4 Observation_SWC1 - Server Ports C/S Interface for NvM_ReadBlock for NvBlockDescriptor4 NvBlockSwComponent : ServerPort Observation_SWC1 : ClientPort	
Summary	To check the NvDataHandling for Explicit Sender Receiver Communication for Primitive Data Type if dirtyflag mechanism is disabled. Control_SWC has to invoke client-server operation (NvM_WriteBlock) for NvBlockDescriptor4 to complete the write operation in NvBlock. Observation_SWC1 invokes NvM_ReadBlock for NvBlockDescriptor4. Wait until JobEndNotification from NvM. Observation_SWC1 does the explicit read on its require port.The read data will have latest value which is written by Control_SWC	
Needed Adaptation to other Releases	For Autosar 4.0.3, instead of PRPortPrototype, needs to configure PPort and RPort separately for NvBlockDescriptor	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Run_Ctrl_6 starts	
Step 2	**[RUN<Run_Ctrl_6>]** Run_Ctrl_6 executes Rte_Write on VDP_Nv6 with value as below: App_Data7 = 0x1A1A	**[RUN<Run_Ctrl_6>]** Rte_Write should return RTE_E_OK
Step 3	**[CP]** Run_Ctrl_6 terminates	
Step 4	**[CP]** Run_Ctrl_WriteBlock_4 starts	
Step 5	**[RUN<Run_Ctrl_WriteBlock_4>]** Run_Ctrl_WriteBlock_4 executes Rte_Call of NvM_WriteBlock for Block_4	**[RUN<Run_Ctrl_WriteBlock_4>]** NvM_WriteBlock should return E_OK
Step 6	**[CP]**	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of

	Wait until JobEndNotification (NvM_WriteBlock) from NvM Check call for NvMNotifyJobFinished	Run_Obs1_JobFinished_4 has been called for NvBlockDescriptor4 with ServiceId value 0x7 and job_result NVM_REQ_OK.
Step 7	**[CP]** Run_Obs1_ReadBlock_4 starts	
Step 8	**[RUN<Run_Obs1_ReadBlock_4>]** Run_Obs1_ReadBlock_4 executes Rte_Call of NvM_ReadBlock() for NvBlockDescriptor4	**[RUN<Run_Obs1_ReadBlock_4>]** NvM_ReadBlock should return E_OK
Step 9	**[CP]** Wait until JobEndNotification (NvM_ReadBlock) from NvM Check call for NvMNotifyJobFinished	**[SWC<Observation_SWC1>]** NvMNotifyJobFinished server operation of Run_Obs1_JobFinished_4 has been called for NvBlockDescriptor4 with ServiceId value 0x6 and job_result NVM_REQ_OK.
Step 10	**[CP]** Run_Obs1_6 starts	
Step 11	**[RUN<Run_Obs1_6>]** Run_Obs1_6 executes Rte_Read from VDP_Nv6	**[RUN<Run_Obs1_6>]** Read data is as below: Nv_Data6 = 0x1A1A
Post-conditions	None	

8 Miscellaneous features

8.1 General Test Objective and Approach

This test suite provides additional test cases for miscellaneous features of the RTE, when they do not require a complete test suite on their own.

8.1.1 Test System

8.1.1.1 Overview on Architecture

The basic test setup is depicted in Figure 9. Test cases require a SW-C on the SUT, which uses the tested features. This SW-C may need multiple PortPrototypes and RunnableEntities, which re described in each test case's "Configuration Parameters" field. A result may be observed on the bus.

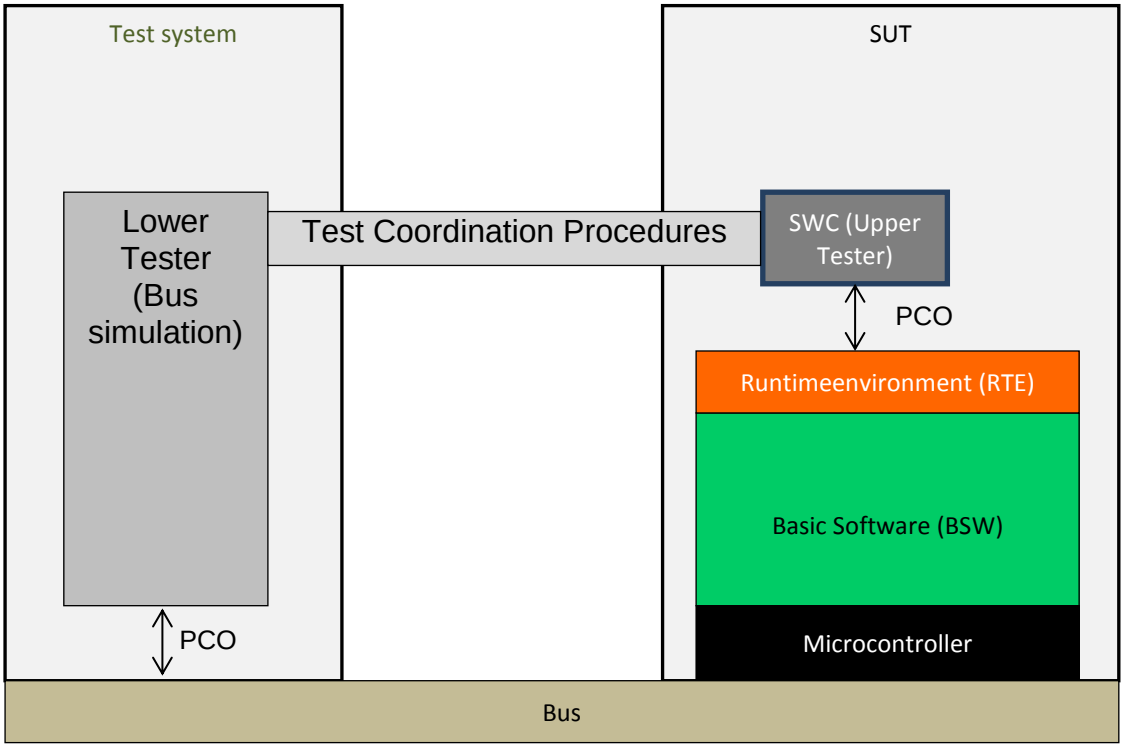


Figure 9: Test Setup for Inter-ECU Test

8.1.1.2 Specific Requirements

None

8.1.1.3 Test Coordination Requirements

Test Coordination Procedures are needed to synchronize the runnables of the SWCs or to wait for different invocation instances of runnables.

TCPs are also needed to collect the test results of the SWCs and the Bus simulation at one central place in order to derive the test verdict.

It is up to the test system designer/implementer to define that "central place" and to design/implement the test coordination functionality.

8.1.2 Test Configuration

The configuration required to implement and execute the test cases is described in the “Configuration Parameters” field of each test case.

These requirement on configuration describe

- The needed configuration for the description of the SW-C associated to the test case
- The needed configuration for the EcuExtract associated to the test case

8.2 Re-usable Test Steps

None.

8.3 Test Cases

8.3.1 [ATS_RTE_00691] InterRunnableVariables With Explicit Behavior

Test Objective	InterRunnableVariables With Explicit Behavior		
ID	ATS_RTE_00691	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01305 RTE: SWS_Rte_01306 RTE: SWS_Rte_03560 RTE: SWS_Rte_03565 RTE: SWS_Rte_03567 RTE: SWS_Rte_03580		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * VariableDataPrototype_TC59 in the role explicitInterRunnableVariable - with initValue configured (Ex: 0x05) * A RunnableEntity to execute the test sequence - with a readLocalVariable which references VariableDataPrototype_TC59 - with a writtenLocalVariable which references VariableDataPrototype_TC59		
Summary	The test checks for explicit access to IRV the value read for an IRV before it is written and checks that an IRV value written explicitly is available immediately for reading by executing Rte_IrvRead in the same execution-instance as the Rte_IrvWrite call.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**

Step 1	[SWC] Invoke Rte_IrvRead to read VariableDataPrototype_TC59.	[SWC] Rte_IrvRead returns the initValue
Step 2	[SWC] Invoke Rte_IrvWrite to write VariableDataPrototype_TC59 with a value different from the initValue.	-
Step 3	[SWC] Invoke Rte_IrvRead to read VariableDataPrototype_TC59.	[SWC] Rte_IrvRead returns the data which was written through Rte_IrvWrite.
Post-conditions	NONE	

8.3.2 [ATS_RTE_00692] InterRunnableVariables With Implicit Behavior

Test Objective	InterRunnableVariables With Implicit Behavior		
ID	ATS_RTE_00692	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01303 RTE: SWS_Rte_01304 RTE: SWS_Rte_03550 RTE: SWS_Rte_03553 RTE: SWS_Rte_03580 RTE: SWS_Rte_03582 RTE: SWS_Rte_03584 RTE: SWS_Rte_07022		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * A PPortPrototype and an RPortPrototype typed by a ClientServerInterface with one operation Operation_TC60 - these ports are connected * VariableDataPrototype_TC60 in the role implicitInterRunnableVariable - with initValue configured (Ex: 0x05) * RunnableEntity_TC60 to execute the test sequence - with a readLocalVariable which references VariableDataPrototype_TC60 - with a writtenLocalVariable which references VariableDataPrototype_TC60 - with a SynchronousServerCallPoint * RunnableEntity_TC60Server - started by an OperationInvokedEvent which references Operation_Tc60 - with a readLocalVariable which references VariableDataPrototype_TC60		
Summary	The test checks for implicit access to IRV the value read for an IRV before it is written and checks that an IRV value written implicitly is available only after the end of the execution-instance.		

Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[RUN<RunnableEntity_TC60>] Invoke Rte_IrvIRead to read VariableDataPrototype_TC60.	[RUN<RunnableEntity_TC60>] Rte_IrvIRead returns initValue.
Step 2	[RUN<RunnableEntity_TC60>] Invoke Rte_IrvIWrite to write VariableDataPrototype_TC60 initValue+1.	-
Step 3	[RUN<RunnableEntity_TC60>] Invoke Rte_IrvIRead to read VariableDataPrototype_TC60.	[RUN<RunnableEntity_TC60>] Rte_IrvIRead returns initValue+1.
Step 4	[RUN<RunnableEntity_TC60>] Invoke Operation_TC60	[RUN<RunnableEntity_TC60Server>] RunnableEntity_TC60Server is invoked.
Step 5	[RUN<RunnableEntity_TC60Server>] Invoke Rte_IrvIRead to read VariableDataPrototype_TC60.	[RUN<RunnableEntity_TC60Server>] Rte_IrvIRead returns initValue.
Step 6	[RUN<RunnableEntity_TC60Server>] Return to exit from this RunnableEntity_TC60 execution-instance.	[RUN<RunnableEntity_TC60>] Rte_Call returns and RunnableEntity_TC60 resumes its execution.
Step 7	[RUN<RunnableEntity_TC60>] Return to exit from this RunnableEntity_TC60 execution-instance.	-
Step 8	[CP] Wait for the next execution-instance of RunnableEntity_TC60.	[RUN<RunnableEntity_TC60>] RunnableEntity_TC60 is invoked.
Step 9	[RUN<RunnableEntity_TC60>] [SWC] Invoke Rte_IrvIRead to read VariableDataPrototype_TC60.	[RUN<RunnableEntity_TC60>] Rte_IrvIRead returns initValue+1.
Step 10	[RUN<RunnableEntity_TC60>] Invoke Operation_TC60	[RUN<RunnableEntity_TC60Server>] RunnableEntity_TC60Server is invoked.
Step 11	[RUN<RunnableEntity_TC60Server>] Invoke Rte_IrvIRead to read VariableDataPrototype_TC60.	[RUN<RunnableEntity_TC60Server>] Rte_IrvIRead returns initValue+1.
Post-conditions	NONE	

8.3.3 [ATS_RTE_00693] Enhanced Rte_Mode API Functionality

Test Objective	Enhanced Rte_Mode API Functionality		
ID	ATS_RTE_00693	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document	
Trace to SWS Item	RTE: SWS_Rte_02512 RTE: SWS_Rte_02631 RTE: SWS_Rte_02632 RTE: SWS_Rte_02667 RTE: SWS_Rte_02669 RTE: SWS_Rte_02674 RTE: SWS_Rte_07173 RTE: SWS_Rte_08500 RTE: SWS_Rte_08501 RTE: SWS_Rte_08504 RTE: SWS_Rte_08505
Requirements / Reference to Test Environment	
Configuration Parameters	A ModeDeclarationGroup MDG_TC67 with * 2 ModeDeclarations: StopMode=0 EndMode=1 * initialMode=StopMode A SW-C with * A PPortPrototype which - is typed with a ModeSwitchInterface with a ModeDeclarationGroupPrototype with type=MDG_TC67 - has a ModeSwitchSenderComSpec with enhancedModeApi=TRUE * An RPortPrototype which - is typed by the same ModeSwitchInterface - has a ModeSwitchReceiverComSpec with enhancedModeApi=TRUE - is connected with the previous PPortPrototype * RunnableEntity_TC67 to initiate the test sequence - with a modeSwitchPoint which references the ModeDeclarationGroupPrototype of the PPortPrototype - with a modeAccessPoint which references the ModeDeclarationGroupPrototype of the RPortPrototype * RunnableEntity_OnEntryEndMode which - is started by a SwcModeSwitchEvent (activation=onEntry mode=EndMode) - has a modeAccessPoint which references the ModeDeclarationGroupPrototype of the RPortPrototype
Summary	This test case verifies the enhanced Rte_Mode API. When an application SW-C calls 'Enhanced Rte_Mode' API the RTE provides the currently active mode of the requested mode switch port and if the mode machine instance is in transition then additionally the values of the previous mode and the next mode are provided.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state
Main Test Execution	
Test Steps	**Pass Criteria**

Step 1	[RUN<RunnableEntity_TC67>] Invoke Rte_Mode(&PreviousMode, &NextMode).	[RUN<RunnableEntity_TC67>] Rte_Mode returns RTE_MODE_MDG_TC67_StopMode and the value in the OUT parameters are: PreviousMode=RTE_MODE_MDG_TC67_StopMode NextMode=RTE_MODE_MDG_TC67_StopMode
Step 2	[RUN<RunnableEntity_TC67>] Invoke Rte_Switch(RTE_MODE_MDG_TC67_EndMode)	[RUN<RunnableEntity_TC67>] Rte_Switch returns RTE_E_OK.
Step 3	-	[RUN<RunnableEntity_OnEntryEndMode>] RunnableEntity_OnEntryEndMode is invoked.
Step 4	[RUN<RunnableEntity_OnEntryEndMode>] Invoke Rte_Mode(&PreviousMode, &NextMode).	[RUN<RunnableEntity_OnEntryEndMode>] Rte_Mode returns RTE_TRANSITION_MDG_TC67 and the value in the OUT parameters are: PreviousMode=RTE_MODE_MDG_TC67_StopMode NextMode=RTE_MODE_MDG_TC67_EndMode
Step 5	[CP] Wait for the next execution-instance of RunnableEntity_TC67	[RUN<RunnableEntity_TC67>] RunnableEntity_TC67is invoked.
Step 6	[RUN<RunnableEntity_TC67>] Invoke Rte_Mode(&PreviousMode, &NextMode).	[RUN<RunnableEntity_TC67>] Rte_Mode returns RTE_MODE_MDG_TC67_EndMode and the value in the OUT parameters are: PreviousMode=RTE_MODE_MDG_TC67_EndMode NextMode=RTE_MODE_MDG_TC67_EndMode
Step 7	[RUN<RunnableEntity_TC67>] Invoke Rte_Switch(RTE_MODE_MDG_TC67_EndMode)	[RUN<RunnableEntity_TC67>] Rte_Switch returns RTE_E_OK.
Step 8	-	[RUN<RunnableEntity_OnEntryEndMode>] RunnableEntity_OnEntryEndMode is invoked.
Step 9	[RUN<RunnableEntity_OnEntryEndMode>] Invoke Rte_Mode(&PreviousMode, &NextMode).	[RUN<RunnableEntity_OnEntryEndMode>] Rte_Mode returns RTE_TRANSITION_MDG_TC67 and the value in the OUT parameters are: PreviousMode=RTE_MODE_MDG_TC67_EndMode NextMode=RTE_MODE_MDG_TC67_EndMode
Post-conditions	NONE	

8.3.4 [ATS_RTE_00702] Ports APIs Functionality

Test Objective	Ports APIs Functionality		
ID	ATS_RTE_00702	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01354 RTE: SWS_Rte_01355 RTE: SWS_Rte_02613 RTE: SWS_Rte_02614 RTE: SWS_Rte_02615 RTE: SWS_Rte_02619 RTE: SWS_Rte_03602 RTE: SWS_Rte_03603		
Requirements / Reference to Test Environment			
Configuration Parameters	2 SenderReceiverInterfaces: IF_TC92a, IF_TC92b with VariableDataPrototype_TC92a (resp. VariableDataPrototype_TC92), swImplPolicy standard A SW-C with * PPortPrototype P_TC92a1, typed by IF_TC92a * PPortPrototype P_TC92a2, typed by IF_TC92a * PPortPrototype P_TC92b1, typed by IF_TC92b * PPortPrototype P_TC92b2, typed by IF_TC92b * Referenced by PortAPIOption, indirectAPI=TRUE, except P_TC92b2: indirectAPI=FALSE * RPortPrototype R_TC92a1, typed by IF_TC92a, connected to P_TC92a1 * RPortPrototype P_TC92a2, typed by IF_TC92a, connected to P_TC92a2 * RPortPrototype P_TC92b1, typed by IF_TC92b, connected to P_TC92b1 * RPortPrototype P_TC92b2, typed by IF_TC92b, connected to P_TC92b2 * With NonqueuedReceiverComSpec, initValue=0 * RunnableEntity to execute the test sequence - with a 4 dataSendPoints for the dataElements in P_TC92a1, P_TC92a2, P_TC92b1, P_TC92b2 - with a 4 dataReceivePointByValue for the dataElements in R_TC92a1, R_TC92a2, R_TC92b1, R_TC92b2		
Summary	Rte_Ports API provide an array of the ports of a given interface type and a given provide / require usage that can be accessed by the indirect API. i.e. Rte_Ports API returns array of port data structures of the corresponding interface type and usage. Rte_NPorts API provides the number of ports of a given interface type and provide/require usage that can be accessed through the indirect API. i.e. Rte_NPorts API returns number of port data structures of the corresponding interface type and usage. Rte_Port API provides access to the port data structure for a single port of a particular software component instance. This allows a software component to extract		

	a sub-group of ports characterized by the same interface in order to iterate over this sub-group.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_Ports_IF_TC92a_P.	[SWC] Rte_Ports returns an array of PortHandles != NULL_PTR.
Step 2	[SWC] Invoke Rte_NPorts_IF_TC92a_P.	[SWC] Rte_NPorts returns 2.
Step 3	[SWC] For the 2 port handles returned by Rte_Ports invoke the function pointer named Write_VariableDataPrototype_TC92a in the port handle with value 1.	[SWC] Each function call returns RTE_E_OK.
Step 4	[SWC] Invoke Rte_DRead_R_TC92a1_VariableDataPrototype_TC92a	[SWC] Rte_DRead returns 1
Step 5	[SWC] Invoke Rte_DRead_R_TC92a2_VariableDataPrototype_TC92a	[SWC] Rte_DRead returns 1
Step 6	[SWC] Invoke Rte_Ports_IF_TC92b_P.	[SWC] Rte_Ports returns an array of ports with address != NULL_PTR.
Step 7	[SWC] Invoke Rte_NPorts_IF_TC92b_P.	[SWC] Rte_NPorts returns 1.
Step 8	[SWC] For the port handle returned by Rte_Ports invoke the function pointer named Write_VariableDataPrototype_TC92b in the port handle with value 1.	[SWC] The function call returns RTE_E_OK.
Step 9	[SWC] Invoke Rte_DRead_R_TC92b1_VariableDataPrototype_TC92b	[SWC] Rte_DRead returns 1
Step 10	[SWC] Invoke Rte_DRead_R_TC92b2_VariableDataPrototype_TC92b	[SWC] Rte_DRead returns 0
Step 11	[SWC] Invoke Rte_Port_P_TC92b2.	[SWC] Rte_Port returns a PortHandle!=NULL_PTR.
Step 12	[SWC] For the port handle returned by Rte_Port invoke the function pointer named Write_VariableDataPrototype_TC92b in the port handle with value 2.	[SWC] The function call returns RTE_E_OK.
Step 13	[SWC] Invoke Rte_DRead_R_TC92b1_VariableDataPrototype_TC92b	[SWC] Rte_DRead returns 2

Post-conditions	NONE

8.3.5 [ATS_RTE_00703] Rte_Prm API Functionality

Test Objective	Rte_Prm API Functionality		
ID	ATS_RTE_00703	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_03928 RTE: SWS_Rte_03929		
Requirements / Reference to Test Environment			
Configuration Parameters	A ParameterSwComponentType * ParameterDataPrototype_TC97 with initValue = Ex:1 A SW-C with * An RPortPrototype to receive ParameterDataPrototype_TC97 - connected to the ParameterSwComponentType * A RunnableEntity to execute the test sequence - with a parameterAccess which references ParameterDataPrototype_TC97		
Summary	A SW-C is connected to a ParameterSwComponentType. The test case checks that the value of the parameter can be accessed with the Rte_Prm API.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invoke Rte_Prm		[SWC] Rte_Prm returns the parameter's value (Ex: 1).
Post-conditions	NONE		

8.3.6 [ATS_RTE_00704] Rte_CData API For sharedParameter

Test Objective	Rte_CData API For sharedParameter		
ID	ATS_RTE_00704	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed

Trace to Requirement on Acceptance Test Document		
Trace to SWS Item	RTE: SWS_Rte_01252 RTE: SWS_Rte_01300 RTE: SWS_Rte_03927	
Requirements / Reference to Test Environment		
Configuration Parameters	A SW-C with * ParameterDataPrototype_TC98 in the role sharedParameter with initValue=Ex: 1 * A RunnableEntity to execute the test sequence - with a parameterAccess which references ParameterDataPrototype_TC98	
Summary	A SW-C is configured with a ParameterDataPrototype in the role sharedParameter. The test case checks that the value for the parameter can be accessed with the Rte_CData API.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_CData for the sharedParameter ParameterDataPrototype_TC98.	[SWC] Rte_CData returns the initValue of ParameterDataPrototype_TC98.
Post-conditions	NONE	

8.3.7 [ATS_RTE_00705] Rte_CData API For perInstanceParameter

Test Objective	Rte_CData API For perInstanceParameter		
ID	ATS_RTE_00705	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01252 RTE: SWS_Rte_01300 RTE: SWS_Rte_03952		
Requirements / Reference to Test Environment			
Configuration Parameters	A SW-C with * ParameterDataPrototype_TC99 in the role perInstanceParameter with initValue=Ex: 1		

	* A RunnableEntity to execute the test sequence - with a parameterAccess which references ParameterDataPrototype_TC99	
Summary	A SW-C is configured with a ParameterDataPrototype in the role perInstanceParameter. The test case checks that the value for the parameter can be accessed with the Rte_CData API.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Invoke Rte_CData for the perInstanceParameter ParameterDataPrototype_TC99	[SWC] Rte_CData returns the initValue of ParameterDataPrototype_TC99.
Post-conditions	NONE	

8.3.8 [ATS_RTE_00706] Rte_Pim API Functionality

Test Objective	Rte_Pim API Functionality		
ID	ATS_RTE_00706	**AUTOSAR Releases**	4.0.3|4.2.1|4.2.2
Affected Modules	RTE	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	RTE: SWS_Rte_01118 RTE: SWS_Rte_01119 RTE: SWS_Rte_01299		
Requirements / Reference to Test Environment			
Configuration Parameters	An ApplicationSwComponentType with * attribute supportsMultipleInstantiation=TRUE * VariableDataPrototype_TC100 in the role arTypedPerInstanceMemory with initValue=Ex: 1 * RunnableEntity_TC100 to execute the test sequence This ApplicationSwComponentType is instantiated twice.		
Summary	The test case checks that the Rte_Pim API provides access to different PerInstanceMemory in case an ApplicationSwComponentType is instantiated multiple times.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[CP] Wait for the execution of RunnableEntity_TC100 for the first instance of the ApplicationSwComponentType.	[RUN<RunnableEntity_TC100>] RunnableEntity_TC100 is invoked.
Step 2	[SWC] Invoke Rte_Pim_VariableDataPrototype_TC100 for the Rte_Instance received in step 1.	[RUN<RunnableEntity_TC100>] Rte_Pim returns the reference to a PerInstanceMemory != NULL_PTR.
Step 3	[CP] Wait for the execution of RunnableEntity_TC100 for the other instance of the ApplicationSwComponentType.	[RUN<RunnableEntity_TC100>] RunnableEntity_TC100 is invoked with a Rte_Instance argument that differ from the one in step 1.
Step 4	[SWC] Invoke Rte_Pim_VariableDataPrototype_TC100 for the instance received in step 3.	[RUN<RunnableEntity_TC100>] Rte_Pim returns the reference to a PerInstanceMemory != NULL_PTR that differs from the address returned by Rte_Pim in step 2.
Post-conditions	NONE	