

Document Title	System Tests of Adaptive Platform
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	890

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	18-10

Document Change History			
Date	Release	Changed by	Description
2018-10-31	18-10	AUTOSAR Release Management	<ul style="list-style-type: none"> Added RS traceability for test cases Added ISO 9646 framework and mapping on system test architecture Added more test cases for CM, REST, EMO, and UCM
2018-03-31	18-03	AUTOSAR Release Management	<ul style="list-style-type: none"> Test case for RESTful communication is added Test case for Security is added Test case for Update and Configuration Management is added Test case for E2E is added
2010-10-27	17-10	AUTOSAR Release Management	<ul style="list-style-type: none"> Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Acronyms and abbreviations	6
2	Scope of Document	7
2.1	Overview on test architecture	7
3	Limitations	9
4	Test configuration and test steps for Communication Management	10
4.1	Test System	10
4.1.1	Test configurations Communication Management	10
4.1.2	Test configurations REST	11
4.2	Test cases	12
4.2.1	[STS_CM_00001] Local and remote service discovery.	12
4.2.2	[STS_CM_00002] Communication for Methods.	13
4.2.3	[STS_CM_00003] Communication for Events based on polling-based style.	15
4.2.4	[STS_CM_00004] Communication for Events based on event-based style.	17
4.3	Test cases REST	19
4.3.1	[STS_REST_00001] Client in backend/ cloud and server in vehicle communicates according to REST	19
4.3.2	[STS_REST_00002] Client in vehicle and server in backend/ cloud communicates according to REST	23
4.3.3	[STS_REST_00003] Portability of RESTful adaptive applications	26
4.3.4	[STS_REST_00004] Data Representation	27
4.3.5	[STS_REST_00005] Event communication with Web-sockets	29
5	Test configuration and test steps for Execution Management	31
5.1	Test System	31
5.1.1	Test configurations	31
5.1.1.1	STC_EMO_00001	31
5.1.1.2	STC_EMO_00002	32
5.1.1.3	STC_EMO_00003	33
5.2	Test cases	34
5.2.1	[STS_EMO_00001] Startup of applications with change of machine state.	34
5.2.2	[STS_EMO_00002] Shutdown of applications with change of machine state to Shutdown	36
5.2.3	[STS_EMO_00003] Ordered Startup and Shutdown of Executables based on the dependency with other processes	37
5.2.4	[STS_EMO_00004] Startup of applications with change of Function Group state	39

5.2.5	[STS_EMO_00005] Execution Management shall prevent Processes from directly starting other Processes	40
6	Test configuration and test steps for Diagnostics	42
6.1	Test System	42
6.1.1	Test configurations	42
6.2	Test cases	43
6.2.1	[STS_DIAG_00001] Utilization of Diagnostic service Read DataByIdentifier (0x22) by external Tester via UDS messages over DoIP.	43
6.2.2	[STS_DIAG_00002] Utilization of Diagnostic service Routine Control (0x31) by external Tester via UDS messages over Do IP.	44
6.2.3	[STS_DIAG_00003] Utilization of Diagnostic service Tester Present (0x3E) by External Tester via UDS messages over DoIP.	45
6.2.4	[STS_DIAG_00004] Utilization of Diagnostic service Write DataByIdentifier (0x2E) by External Tester via UDS messages over DoIP.	47
7	Test configuration and test steps for Logging and Tracing	49
7.1	Test System	49
7.1.1	Test configurations	49
7.2	Test cases	50
7.2.1	[STS_LT_00001] Receiving of log messages from LT module by external Tester and remote control of application's default log level.	50
7.2.2	[STS_LT_00002] Receiving of log messages from LT modules of several ECUs.	51
8	Test configuration and test steps for Persistency	53
8.1	Test System	53
8.1.1	Test configurations	53
8.2	Test cases	54
8.2.1	[STS_PER_00001] Storing an integer in a key-value database.	54
8.2.2	[STS_PER_00002] Storing a float in a key-value database.	54
8.2.3	[STS_PER_00003] Storing a string in a key-value database.	55
8.2.4	[STS_PER_00004] Storing a string in a file.	56
8.2.5	[STS_PER_00005] Storing an integer in a key-value database and retrieving it after reboot.	56
8.2.6	[STS_PER_00006] Storing a string in a file and retrieving it after reboot.	57
9	Test configuration and test steps for Security	59
9.1	Test System	59
9.1.1	Test configurations	59
9.2	Test cases	60

9.2.1	[STS_SEC_00001] Rejecting local service usage by an unauthorized application	60
9.2.2	[STS_SEC_00002] Rejecting events sent by an unauthorized application	61
9.2.3	[STS_SEC_00003] Rejecting events if no application is authorized to receive them	62
10	Test configuration and test steps for Update and Configuration Management	64
10.1	Test System	64
10.1.1	Test configurations	64
10.2	Test cases	65
10.2.1	[STS_UCM_00001] Check, if an update of a SW package is available.	65
10.2.2	[STS_UCM_00002] Update a SW package, on user request.	66
10.2.3	[STS_UCM_00003] Installing a SW package on user approval.	67
10.2.4	[STS_UCM_00004] Uninstalling a SW package, on user request.	68
10.2.5	[STS_UCM_00005] Rollback to previous version, after corrupted SW package installation.	69
10.2.6	[STS_UCM_00006] Read update history on an adaptive platform, on demand.	70
10.2.7	[STS_UCM_00007]Data Transfer from Multiple clients,Simultaneously.	71
10.2.8	[STS_UCM_00008]Install/Update/Removal of SW Package from multiple clients,sequentially.	72
10.2.9	[STS_UCM_00009]Cancel Install/Update operation of SW Package	73
11	Test configuration and test steps for E2E Protection	75
11.1	Test System	75
11.1.1	Test configurations E2E Protection	75
11.2	Test cases	76
11.2.1	[STS_E2E_00001] E2E Protection from AP to AP	76
11.2.2	[STS_E2E_00002] Corrupting APP Affecting Communication	78
12	References	81

1 Acronyms and abbreviations

The glossary below includes terms, acronyms and abbreviations relevant to System Test Specification that are not included in the AUTOSAR Glossary [1].

Abbreviation / Acronym:	Description:
Rx	Reception
RS	Requirement Specification
NRC	Negative Response Code
Tx	Transmission
ST	System Test
SM	State Manager
TCP	Test Coordination Procedures
PCO	Point of Control and Observation
SUT	System Under Test
UT	Upper Tester
IUT	Implementation Under Test
LT	Lower Tester
UTA	UCM Test Application

2 Scope of Document

The system test cases are used to validate RS items in order to confirm whether requirements of functional cluster are satisfied by the AUTOSAR Adaptive Platform Demonstrator. Each test case is applicable with the coupled specification release.

In this R18-10 release, Requirement Specification of CM (someip, REST), EMO, DIA, LT, PER, IAM, UCM and E2E is in the scope of this document.

2.1 Overview on test architecture

In this section, System Test architecture is described according to ISO 9646 test architecture manner. In System Test, FC tester is called as LT (Lower Tester) which stimulate and observe IUT (Implementation Under Test) behavior. AP instances is called as IUT (Implementation Under Test) which is the test target. Applications is called as UT (Upper Tester) which is stimulated by LT and take an action to request test step (e.g. sending message) to IUT.

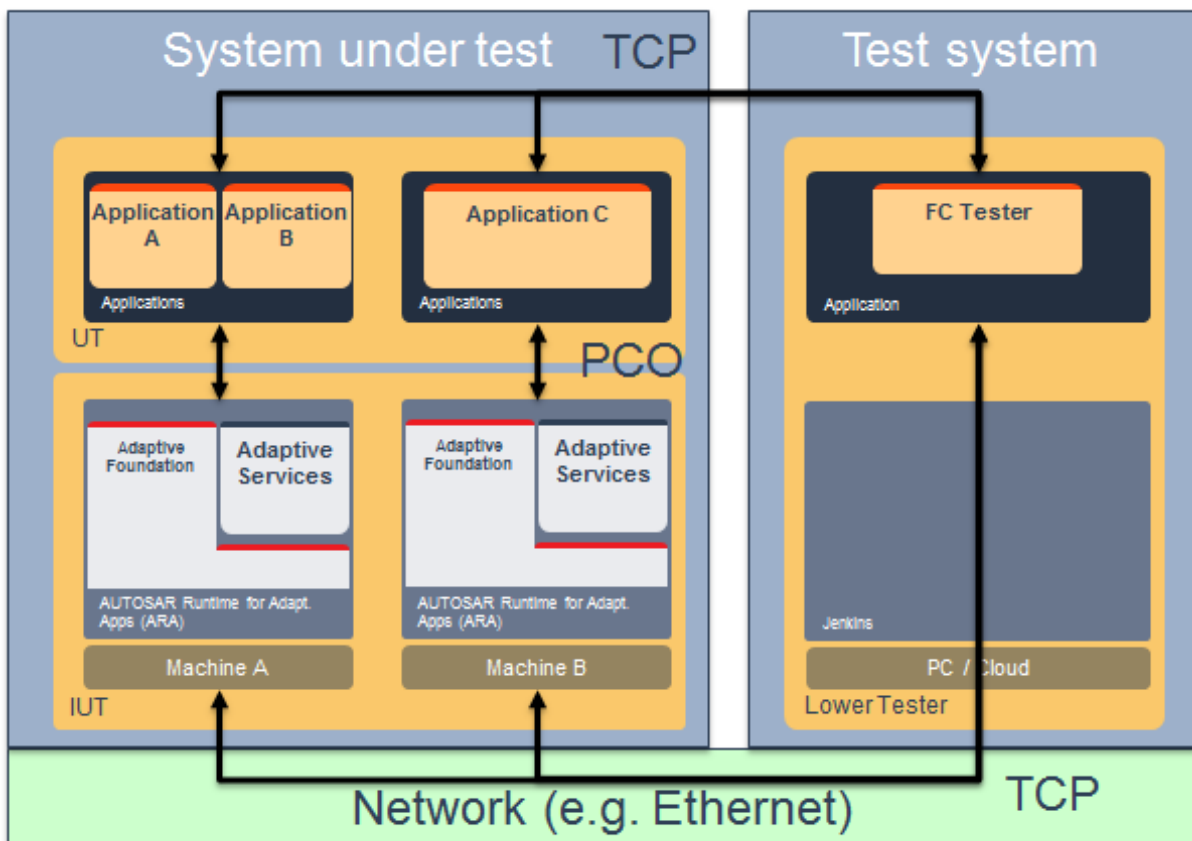


Figure 2.1: System Test architecture

The following picture describing that mapping to System Test implementation. In ST demonstrator, TCP is realized by stimulating application via Diagnostics routine ser-

vice. PCO is realized by requesting action via ARA::API, and receive/ transmit Ethernet message so that IUT could react. Application send message after certain step is passed so that test system could observe what happens on System under test.

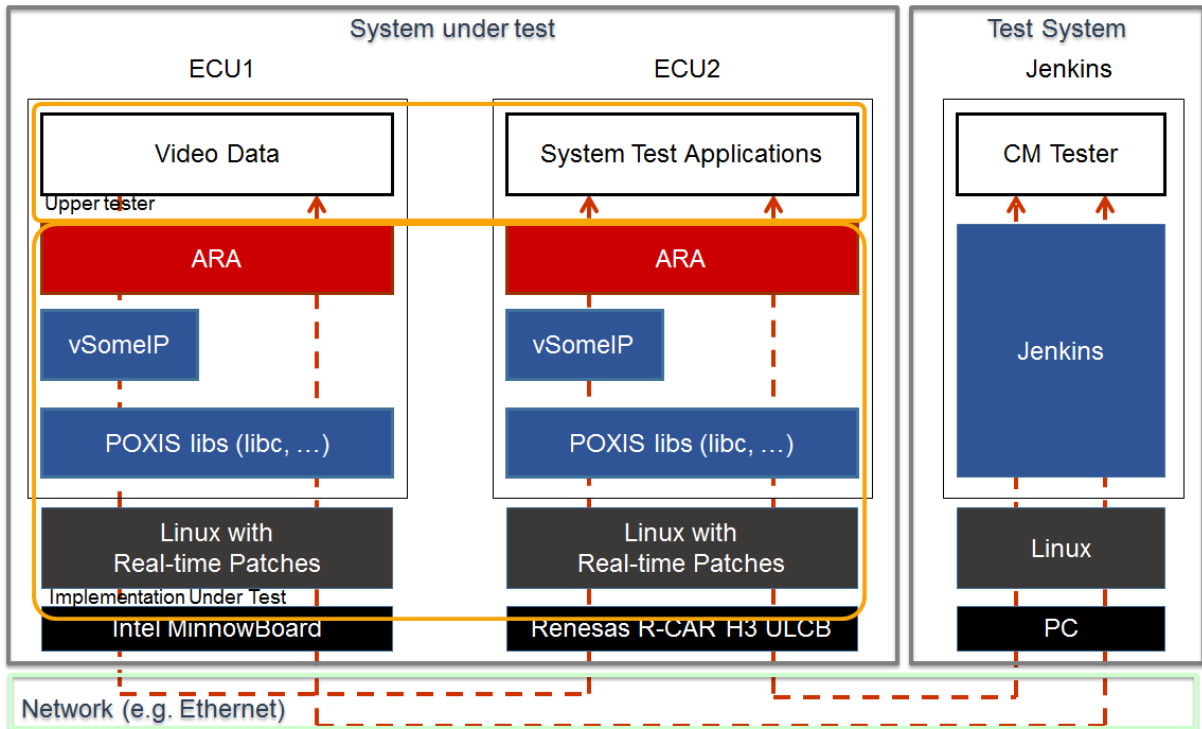


Figure 2.2: Map to System Test implementation

3 Limitations

There are several limitations in this document.

- Test cases may not cover whole RS as specified against test cases
- Test setup figure may not exactly reflect the test configuration
- Test cases may not be fully covered by corresponding system test implementations
- System test cases are just examples, since there could be many ways to define and implement use case scenarios
- DIAG does not have any RS traceability, as it is intended to reuse WP-T results
- LT does not have any RS traceability. Traceability will be added in next release

4 Test configuration and test steps for Communication Management

4.1 Test System

4.1.1 Test configurations Communication Management

Configuration ID	STC_CM_00001
Description	Standard Jenkins server for Communication Management test
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

Configuration ID	STC_CM_00002
Description	Scenario 2 Variant 2 - Reference Deployment
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Communication Management test ([CM Tester]) is connected via Ethernet to [ECU1] hosting the System Test Application [APP1] (as well as [APP4] on the alternative configuration) and [ECU2] hosting the System Test Applications [APP2], [APP3], [APP4] and [APP5].

The [CM Tester] is supposed to collect the results.

The communication between [CM Tester] and the applications on the ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

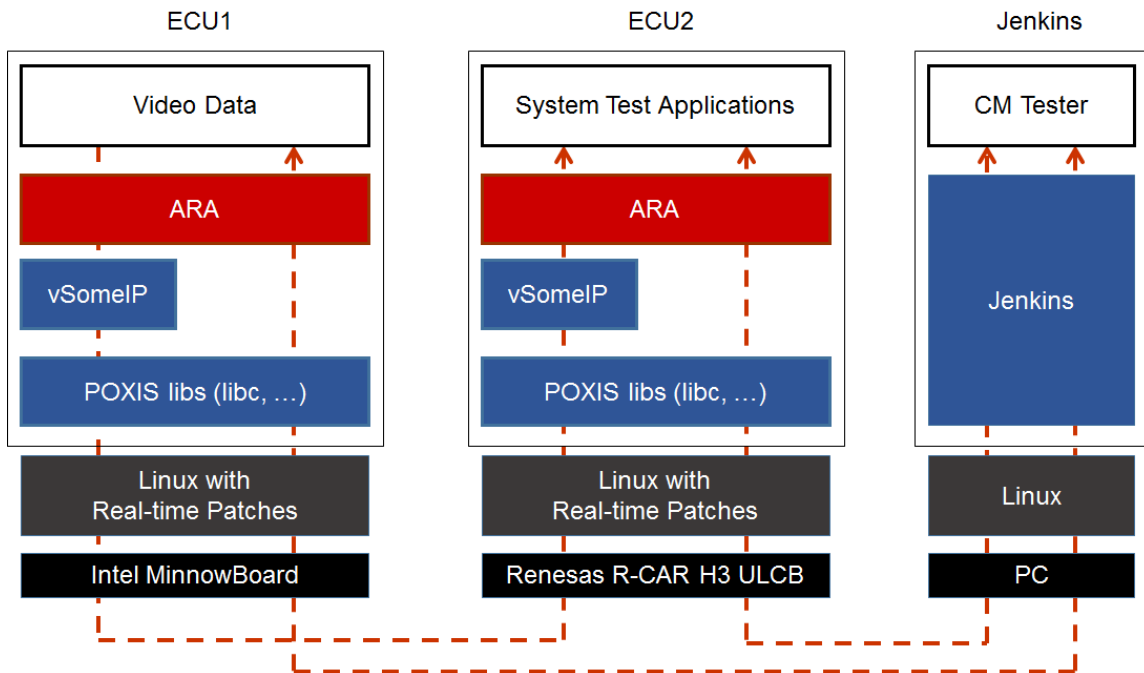


Figure 4.1: Illustration of test setup for Communication Management

4.1.2 Test configurations REST

Configuration ID	STC_REST_00001
Description	Client in backend/ cloud and server in vehicle communicates as per REST
ECU	Intel MinnowBoard Turbot, 192.168.100.5
Backend/ cloud	Server, 192.168.100.10

Configuration ID	STC_REST_00002
Description	Client in vehicle and server in backend/ cloud communicates as per REST
ECU	Intel MinnowBoard Turbot, 192.168.100.5
Backend/ cloud	Client, 192.168.100.10

The Jenkins Server, running the job with the RESTful Communication test [REST Tester] is connected via Ethernet to ECU and backend/ cloud hosting the System Test Applications.

The [REST Tester] is supposed to collect the results.

The communication between [REST Tester] and the applications on the ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

4.2 Test cases

4.2.1 [STS_CM_00001] Local and remote service discovery.

Test Objective	To verify that the applications are able to offer, request and stop services and that service discovery works, establishing the correct communication paths.		
ID	STS_CM_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Communication Management	State	Draft
Trace to RS Criteria	RS_CM_00101, RS_CM_00102, RS_CM_00103, RS_CM_00105, RS_CM_00107, RS_CM_00211.		
Reference to Test Environment	STC_CM_00001		
Configuration Parameters	<ul style="list-style-type: none"> - The existing communication services comprise the following (service names are arbitrary): - [SERVICE1]: Offered by [APP2], requested by [APP1]. - [SERVICE2]: Offered by [APP2], requested by [APP3]. - [SERVICE3]: Offered by [APP1], requested by [APP2]. - [SERVICE4]: Not available, requested by [APP3]. - [SERVICE1], [SERVICE2], [SERVICE3] and [SERVICE4] are attributes of Methods, Events and Fields. 		
Summary	<p>First, the [APP2] and [APP3] applications on [ECU2] are started when Machine State for [ECU2] is changed to Driving.</p> <p>The [APP2] offers the services [SERVICE1] and [SERVICE2] and requests the service [SERVICE3]. [APP3] requests the service [SERVICE2].</p> <p>The [CM Tester] trigger application [APP2] to Stop Offering service [SERVICE2].</p> <p>Then [APP2] again offer service [SERVICE2] and initial reconnection is established between [APP2] and [APP3].</p> <p>Then the [APP1] application on [ECU1] is started when Machine State for [ECU1] is changed to Driving.</p> <p>The [APP1] offers the service [SERVICE3] and requests the service [SERVICE1]. [APP3] requests the service [SERVICE4].</p> <p>The [APP1] stops offering service [SERVICE3]. All services are supposed to be found once available. If a service is not available, the requesting application is expected to have the possibility to assess the availability. Note: As for order of offering, no particular order of offering and requesting is necessary.</p>		
Pre-conditions	<ul style="list-style-type: none"> - [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP1] on [ECU1] and [APP2], [APP3] on [ECU2] are shut down according to Machine State. 		
Post-conditions	CM Tester is disconnected to both ECUs.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[CM Tester] Request change of Machine State to Driving for [ECU2].		Machine State for [ECU2] is changed to Driving.
Step 2	[APP2] Offer service [SERVICE1].		
Step 3	[APP2] Offer service [SERVICE2].		





Step 4	[APP3] Request service [SERVICE2].	Service discovery callback with a handle for service [SERVICE2] is received by [APP3].
Step 5	[CM Tester] Trigger Application [APP2] to Stop Offering service [SERVICE2].	
Step 6	[APP2] Offer service [SERVICE2].	Service discovery callback with a handle for service [SERVICE2] is received by [APP3].
Step 7	[APP2] Request service [SERVICE3].	Service is not available.
Step 8	[CM Tester] Request change of Machine State to Driving for [ECU1].	Machine State for [ECU1] is changed to Driving.
Step 9	[APP1] Offer service [SERVICE3].	
Step 10	[APP2] Request service [SERVICE3].	Service discovery callback with a handle for service [SERVICE3] received by [APP2].
Step 11	[APP1] Request service [SERVICE1].	Service discovery callback with a handle for service [SERVICE1] is received by [APP1].
Step 12	[APP3] Request service [SERVICE4].	Service is not available.
Step 13	[APP1] Stop offering service [SERVICE3].	
Step 14	[APP2] Request service [SERVICE3]	Service is not available.

4.2.2 [STS_CM_00002] Communication for Methods.

Test Objective	To verify that the applications are able to offer, request and receive services and that communication work in a one-to-n communication topology for Methods.		
ID	STS_CM_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Communication Management	State	Draft
Trace to RS Criteria	RS_CM_00101, RS_CM_00102, RS_CM_00211, RS_CM_00212, RS_CM_00213, RS_CM_00214, RS_CM_00215, RS_CM_00225		
Reference to Test Environment	STC_CM_00002		
Configuration Parameters	<ul style="list-style-type: none"> - The existing communication services comprise the following (service names are arbitrary): - [SERVICE5]: Offered by [APP4], requested by [APP5]. - [SERVICE6]: Offered by [APP2], requested by [APP4]. - [SERVICE7]: Offered by [APP3], requested by [APP4]. 		





	<p style="text-align: center;">△</p> <ul style="list-style-type: none"> - [SERVICE5] service receives requested services synchronously. - [SERVICE6] service receives requested services asynchronously. One by querying applications and another by triggering applications. - [SERVICE7] service is an attribute for fire & forget methods. 	
Summary	<p>Firstly the [APP4] application on [ECU1] offers the service [SERVICE5]. This service is requested by one [APP5] instance on [ECU2] and another [APP5] instance on [ECU1].</p> <p>The [APP2] application on [ECU2] offers the service [SERVICE6]. This service is requested by one [APP4] instance on [ECU1].</p> <p>The [APP5] on [ECU2] receives data over service [SERVICE5] from [APP4] as synchronous service call.</p> <p>The [APP5] on [ECU1] receives data over service [SERVICE5] from [APP4] as synchronous service call.</p> <p>The [APP4] receives data as asynchronous service call by querying application [APP2] over service [SERVICE6].</p> <p>Then [APP4] again request service [SERVICE6].</p> <p>The [APP3] application on [ECU2] offers service [SERVICE7]. This service is requested by one [APP4] instance on [ECU1] as fire & forget service call.</p> <p>Then [APP4] receives data over service [SERVICE6] from [APP2] as asynchronous service call by notification.</p> <p>Through successful service discovery, a one-to-n communication topology is established.</p> <p>Note: As for order of offering, no particular order of offering and requesting is necessary.</p>	
Pre-conditions	<ul style="list-style-type: none"> - [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP4], [APP5] on [ECU1] and [APP2], [APP3], [APP5] on [ECU2] are shut down according to Machine State. 	
Post-conditions	CM Tester is disconnected to both ECUs.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[APP4] Offer service [SERVICE5].	
Step 2	[APP5] [ECU2] Request service [SERVICE5].	Service discovery callback with a handle for service [SERVICE5] is received by [APP5] [ECU2].
Step 3	[APP5] [ECU1] Request service [SERVICE5].	Service discovery callback with a handle for service [SERVICE5] is received by [APP5] [ECU1].
Step 4	[APP2] Offer service [SERVICE6].	
Step 5	[APP4] Request service [SERVICE6].	Service discovery callback with a handle for service [SERVICE6] is received by [APP4] [ECU1].
Step 6	[APP5] [ECU2] Receive vehicle data over service [SERVICE5] from [APP4].	[APP5] [ECU2] Data is received from [APP4] over service [SERVICE5].
Step 7	[APP5] [ECU1] Receive vehicle data over service [SERVICE5] from [APP4].	[APP5] [ECU1] Data is received from [APP4] over service [SERVICE5].





Step 8	[APP4] Receive vehicle data over service [SERVICE6].	[APP4] Data is received over service [SERVICE6] by querying application [APP2]
Step 9	[APP4] Request service [SERVICE6].	Service discovery callback with a handle for service [SERVICE6] is received by [APP4] [ECU1].
Step 10	[APP3] Offer service [SERVICE7].	
Step 11	[APP4] Request service [SERVICE7] by fire & forget methods.	Service discovery callback with a handle for service [SERVICE7] may or may not be received by [APP4] [ECU1].
Step 12	[APP4] Receive vehicle data over service [SERVICE6].	[APP4] is notified that the result is available and can be received from application [APP4] over service [SERVICE6].

4.2.3 [STS_CM_00003] Communication for Events based on polling-based style.

Test Objective	To verify that the applications are able to offer, subscribe, receive and stop subscribing services and that communication work in a one-to-n communication topology for Events. The applications are able to receive events and access them in polling-based style.		
ID	STS_CM_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Communication Management	State	Draft
Trace to RS Criteria	RS_CM_00101, RS_CM_00102, RS_CM_00103, RS_CM_00104, RS_CM_00105, RS_CM_00106, RS_CM_00201, RS_CM_00202, RS_CM_00206.		
Reference to Test Environment	STC_CM_00002		
Configuration Parameters	<ul style="list-style-type: none"> - The existing communication services comprise the following (service names are arbitrary): - [SERVICE5]: Offered by [APP4], requested by [APP5]. - Service [SERVICE5] is an attribute of Events. - Reception of services from Server to Proxy is possible using pooling-based style. 		
Summary	<p>The application [APP4] on [ECU2] offers the service [SERVICE5]. This service is subscribed by one application [APP5] instance on [ECU2] and another application [APP5] instance on [ECU1].</p> <p>The application [APP5] [ECU2] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].</p> <p>The application [APP5] [ECU1] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].</p> <p>The application [APP5] [ECU2] polls for receiving events from application [APP4] of service [SERVICE5].</p> <p>The application [APP5] [ECU1] polls for receiving events from application [APP4] of service [SERVICE5].</p> <p>The application [APP4] sends event to subscribed applications [APP5] on [ECU2] and [APP5] on [ECU1] of service [SERVICE5].</p>		





	<p style="text-align: center;">△</p> <p>The application [APP5] [ECU2] receives updated data over service [SERVICE5] from [APP4]. The application [APP5] [ECU1] receives updated data over service [SERVICE5] from [APP4]. The applications [APP5] on [ECU2] and [APP5] on [ECU1] queue received data to ensure it receives the last n events, n being the queue length. The application [APP5] [ECU2] stops subscribing service [SERVICE5]. The application [APP5] [ECU2] monitors state of subscription and gets the current state of subscription. The application [APP5] [ECU1] stops subscribing service [SERVICE5]. The application [APP5] [ECU1] monitors state of subscription and gets the current state of subscription. Through successful service discovery, a one-to-n communication topology is established. Note: As for order of offering, no particular order of offering and requesting is necessary.</p>	
Pre-conditions	<ul style="list-style-type: none"> - [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP4], [APP5] on [ECU2] and [APP5] on [ECU1] are shut down according to Machine State. 	
Post-conditions	CM Tester is disconnected to both ECUs.	
Main Test Execution		
	Test Steps	Pass Criteria
Step 1	[APP4] Offer service [SERVICE5].	
Step 2	[APP5] [ECU2] Subscribe to service [SERVICE5].	
Step 3	[APP5] [ECU1] Subscribe to service [SERVICE5].	
Step 4	[APP5][ECU2] Monitor state of subscription over service [SERVICE5].	[APP5] [ECU2] gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4].
Step 5	[APP5][ECU1] Monitor state of subscription over service [SERVICE5].	[APP5] [ECU1] gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4].
Step 6	[APP5] [ECU2] Poll for receiving events from application [APP4] over service [SERVICE5].	[APP5] [ECU2] Event is not received over service [SERVICE5] of application [APP4].
Step 7	[APP5] [ECU1] Poll for receiving events from application [APP4] over service [SERVICE5].	[APP5] [ECU1] Event is not received over service [SERVICE5] of application [APP4].
Step 8	[APP4] Send event i.e. vehicle data over [SERVICE5] to applications [APP5] [ECU2] and [APP5] [ECU1].	[APP4] Replicates vehicle data send over service [SERVICE5] to applications [APP5] [ECU2] and [APP5] [ECU1].





Step 9	[APP5] [ECU2] Poll for receiving event from application [APP4] over service [SERVICE5].	[APP5] [ECU2] Events received and read them at the same time from service [SERVICE5] of application [APP4].
Step 10	[APP5] [ECU1] Poll for receiving event from application [APP4] over service [SERVICE5].	[APP5] [ECU1] Events received and read them at the same time from service [SERVICE5] of application [APP4].
Step 11	[APP5] [ECU2] Queue received events, n being the queue length.	[APP5] [ECU2] Queue received events until it receives the last n events over service [SERVICE5] of application [APP4].
Step 12	[APP5] [ECU1] Queue received events, n being the queue length.	[APP5] [ECU1] Queue received events until it receives the last n events over service [SERVICE5] of application [APP4].
Step 13	[APP5] [ECU2] Stop subscription of service [SERVICE5].	[APP5] [ECU2] no longer receives events of service [SERVICE5] from [APP4].
Step 14	[APP5] [ECU2] Monitor state of subscription from service [SERVICE5] of application [APP4].	[APP5] [ECU2] gets the current status of subscription, i.e. [APP5] [ECU2] has stopped subscription from service [SERVICE5].
Step 15	[APP5] [ECU1] Stop subscription of service [SERVICE5].	[APP5] [ECU1] no longer receives events of service [SERVICE5] from [APP4].
Step 16	[APP5] [ECU1] Monitor state of subscription from service [SERVICE5] of application [APP4].	[APP5] [ECU1] gets the current status of subscription, i.e. [APP5] [ECU1] has stopped subscription from service [SERVICE5].

4.2.4 [STS_CM_00004] Communication for Events based on event-based style.

Test Objective	To verify that the applications are able to offer, subscribe, monitor, receive and stop subscribing services and that communication work in a one-to-n communication topology for Events. The applications are able to receive events and access them in event-based style.		
ID	STS_CM_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	Communication Management	State	Draft
Trace to RS Criteria	RS_CM_00101, RS_CM_00102, RS_CM_00103, RS_CM_00104, RS_CM_00105, RS_CM_00106, RS_CM_00201, RS_CM_00203, RS_CM_00206.		
Reference to Test Environment	STC_CM_00002		
Configuration Parameters	<ul style="list-style-type: none"> - The existing communication services comprise the following (service names are arbitrary): - [SERVICE5]: Offered by [APP4], requested by [APP5]. - Service [SERVICE5] is an attribute of Events. - Reception of services from Server to Client is possible using event-based style. 		





Summary	<p>The application [APP4] on [ECU2] offers the service [SERVICE5]. This service is subscribed by one application [APP5] instance on [ECU2] and another application [APP5] instance on [ECU1].</p> <p>The application [APP5] [ECU2] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].</p> <p>The application [APP5] [ECU1] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].</p> <p>The application [APP4] sends event to subscribed applications [APP5] on [ECU2] and [APP5] on [ECU1] of service [SERVICE5].</p> <p>The application [APP5] [ECU2] receives updated data over service [SERVICE5] from [APP4].</p> <p>The application [APP5] [ECU1] receives updated data over service [SERVICE5] from [APP4].</p> <p>The applications [APP5] on [ECU2] and [APP5] on [ECU1] queue received events to ensure it receives the last n events, n being the queue length.</p> <p>The application [APP5] [ECU2] stops subscribing service [SERVICE5].</p> <p>The application [APP5] [ECU2] monitors state of subscription and gets the current state of subscription.</p> <p>The application [APP5] [ECU1] stops subscribing service [SERVICE5].</p> <p>The application [APP5] [ECU1] monitors state of subscription and gets the current state of subscription.</p> <p>Through successful service discovery, a one-to-n communication topology is established.</p> <p>Note: As for order of offering, no particular order of offering and requesting is necessary.</p>	
Pre-conditions	<ul style="list-style-type: none"> - [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP4], [APP5] on [ECU2] and [APP5] on [ECU1] are shut down according to Machine State. 	
Post-conditions	CM Tester is disconnected to both ECUs.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[APP4] Offer service [SERVICE5].	
Step 2	[APP5] [ECU2] Subscribe to service [SERVICE5].	
Step 3	[APP5] [ECU1] Subscribe to service [SERVICE5].	
Step 4	[APP5][ECU2] Monitor state of subscription over service [SERVICE5].	[APP5] [ECU2] gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4].
Step 5	[APP5][ECU1] Monitor state of subscription over service [SERVICE5].	[APP5] [ECU1] gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4].
Step 6	[APP4] Send event i.e. vehicle data over [SERVICE5] to applications [APP5] [ECU2] and [APP5] [ECU1].	[APP4] Replicates vehicle data to send over service [SERVICE5] to applications [APP5] [ECU2] and [APP5] [ECU1].
Step 7	[APP5] [ECU2] Get triggered when receiving event over service [SERVICE5] of application [APP4].	[APP5] [ECU2] Events received and read them at the same time from service [SERVICE5].





Step 8	[APP5] [ECU1] Get triggered when receiving event over service [SERVICE5].	[APP5] [ECU1] Events received and read them at the same time from service [SERVICE5] of application [APP4].
Step 9	[APP5] [ECU2] Queue received events, n being the queue length.	[APP5] [ECU2] Queue received events until it receives the last n events over service [SERVICE5] of application [APP4].
Step 10	[APP5] [ECU1] Queue received n events, n being the queue length.	[APP5] [ECU1] Queue received events until it receives the last n events over service [SERVICE5] of application [APP4].
Step 11	[APP5] [ECU2] Stop subscription of service [SERVICE5].	[APP5] [ECU2] no longer receives events of service [SERVICE5] from [APP4]
Step 12	[APP5] [ECU2] Monitor state of subscription from service [SERVICE5] of application [APP4].	[APP5] [ECU2] gets the current status of subscription, i.e. [APP5] [ECU2] has stopped the subscription from service [SERVICE5].
Step 13	[APP5] [ECU1] Stop subscription of service [SERVICE5].	[APP5] [ECU1] no longer receives events of service [SERVICE5] from [APP4]
Step 14	[APP5] [ECU1] Monitor state of subscription from service [SERVICE5] of application [APP4].	[APP5] [ECU1] gets the current status of subscription, i.e. [APP5] [ECU1] has stopped the subscription from service [SERVICE5].

4.3 Test cases REST

4.3.1 [STS_REST_00001] Client in backend/ cloud and server in vehicle communicates according to REST

Test Objective	To verify that server in vehicle responds client-defined request according to REST.		
ID	STS_REST_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	REST	State	Draft
Trace to RS Criteria	RS_CM_00300, RS_CM_00304, RS_CM_00309, RS_CM_00312		
Reference to Test Environment	STC_REST_00001		
Configuration Parameters	RESTful API is configured		
Summary	<ul style="list-style-type: none"> Client is in backend/ cloud and server is in vehicle. First client is set up and request is created with URI and Methods (GET/PUT/ POST/DELETE/OPTIONS). Request is sent and response is received from server. 		





	<p style="text-align: center;">△</p> <ul style="list-style-type: none"> • Server provide a RESTful service [SERVICE1] which has resources [Resource1] and [Resource2]. Each resource has elements like - [Resource1/Element1], [Resource2/Element2]. Element1 have possible states <State1> and <State2> while Element2 have <State3> and <State4>. A new element [Element3] is created in resource [Resource2] using POST and later [Element3] is deleted using DELETE. • Response from server is processed and then client unsubscribe from the event. • Client is stopped. 	
Pre-conditions	<ul style="list-style-type: none"> - [REST Tester] is connected to ECU (vehicle). - ECU is in Machine State Parking. 	
Post-conditions	TCP connections between [REST Tester] and both ECUs are closed.	
Main Test Execution		
	Test Steps	Pass Criteria
Step 1	[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name> ContentLength : <length> Accept: <application/json> Version: HTTP/1.1	
Step 2	[REST Server Application] Server Response: HTTP/1.1 200 OK Content-Type: <application/json> Status: <Status1> URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status>	Positive response is received from Server.
Step 3	[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name> ContentLength : <length> Accept: <application/xml> Version: HTTP/1.1	
Step 4	[REST Server Application] Server Response: HTTP/1.1 200 OK Content-Type: <application/xml> Status: <Status1> URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status>	Positive response is received from Server.





<p>Step 5</p>	<p>[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 6</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <Status1> URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 7</p>	<p>[REST Client Application] Send Request to update Resource1/Element1 (change status 1 to status 2) Method: PUT URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/Status2 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 8</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 9</p>	<p>[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 10</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <Status2> URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 11</p>	<p>[REST Client Application] Send Request to get details of Resource2 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource2 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	



Step 12	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status>	Positive response is received from Server.
Step 13	[REST Client Application] Send Request to create Resorce2/Element3 Method: POST URI: http://<host-name>:<port>/SERVICE1/Resource2/Element3 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 14	[REST Server Application] Server Response: HTTP/1.1 201 Created URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3	Positive response is received from Server.
Step 15	[REST Client Application] Send Request to get details of Resource2 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource2 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 16	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status> URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3/<Status>	Positive response is received from Server.
Step 17	[REST Client Application] Send Request to delete [Element3] Method: DELETE URI: http://<host-name>:<port>/SERVICE1/Resource2/Element3 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 18	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3	Positive response is received from Server.
Step 19	[REST Client Application] Send Request to get details of Resource2 (Element 3 should be deleted) Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource2 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	





Step 20	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status>	Positive response is received from Server.
Step 21	[REST Client Application] Send an invalid URI Request Method = GET, URI: http://<host-name>:<port>/SERVICE5	
Step 22	[REST Server Application] Server replies with Status: 404 URI: http://<host-name>:<port>/SERVICE5	Negative response is received from Server.

4.3.2 [STS_REST_00002] Client in vehicle and server in backend/ cloud communicates according to REST

Test Objective	To verify that server in backend responds client-defined request according to REST.		
ID	STS_REST_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	REST	State	Draft
Trace to RS Criteria	RS_CM_00300, RS_CM_00304, RS_CM_00309, RS_CM_00312		
Reference to Test Environment	STC_REST_00002		
Configuration Parameters	RESTful API is configured		
Summary	<ul style="list-style-type: none"> - Client is in vehicle and server is in backend/ cloud. - First client is set up and request is created with URI and Methods (GET/PUT/ POST/DELETE/OPTIONS). - Request is sent and response is received from server. - Server provide a RESTful service [SERVICE2] which has resources [Resource5] and [Resource6]. Each resource has elements like - [Resource5/Element5], [Resource6/Element6]. Element5 have possible states <State5> and <State6> while Element6 have <State7> and <State8>. A new element [Element7] is created in resource [Resource6] using POST and later [Element7] is deleted using DELETE. - Response from server is processed and then client unsubscribe from the event. Client is stopped. 		
Pre-conditions	<ul style="list-style-type: none"> - [REST Tester] is connected to ECU. - ECU is in Machine State Parking. 		
Post-conditions	TCP connections between [REST Tester] and both ECUs are closed.		
Main Test Execution			
Test Steps			Pass Criteria





<p>Step 1</p>	<p>[REST Client Application] Send Request to get status of Resource5/Element5 Method: GET URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/?Status Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 2</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <Status5> URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 3</p>	<p>[REST Client Application] Send Request to update Resource5/Element5 (change status 5 to status 6) Method: PUT URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/Status6 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 4</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 5</p>	<p>[REST Client Application] Send Request to get status of Resource5/Element5 Method: GET URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/?Status Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	
<p>Step 6</p>	<p>[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <Status6> URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status></p>	<p>Positive response is received from Server.</p>
<p>Step 7</p>	<p>[REST Client Application] Send Request to get details of Resourc6 Method: GET URI: http://<host-name>:<port>/SERVICE2/Resource6 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1</p>	



Step 8	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status>	Positive response is received from Server.
Step 9	[REST Client Application] Send Request to create Resorce6/Element7 Method: POST URI: http://<host-name>:<port>/SERVICE2/Resource6/Element7 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 10	[REST Server Application] Server Response: HTTP/1.1 201 Created URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7	Positive response is received from Server.
Step 11	[REST Client Application] Send Request to get details of Resource6 Method: GET URI: http://<host-name>:<port>/SERVICE2/Resource6 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 12	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status> URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7/<Status>	Positive response is received from Server.
Step 13	[REST Client Application] Send Request to delete [Element7] Method: DELETE URI: http://<host-name>:<port>/SERVICE2/Resource6/Element7 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 14	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7	Positive response is received from Server.





Step 15	[REST Client Application] Send Request to get details of Resource6 Method: GET URI: http://<host-name>:<port>/SERVICE2/Resource6 Host: <host-name> ContentLength : <length> ContentType: <application/json> Version: HTTP/1.1	
Step 16	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status>	Positive response is received from Server.
Step 17	[REST Client Application] Send an invalid URI Request Method = GET, URI: http://<host-name>:<port>/SERVICE5	
Step 18	[REST Server Application] Server replies with Status: 404 URI: http://<host-name>:<port>/SERVICE5	Negative response is received from Server.

4.3.3 [STS_REST_00003] Portability of RESTful adaptive applications

Test Objective	To verify that the same RESTful adaptive application can be used with HTTP/1.1 or a IPC binding without changing any application code.		
ID	STS_REST_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	REST	State	Draft
Trace to RS Criteria	RS_CM_00301		
Reference to Test Environment	STC_REST_00002		
Configuration Parameters	RESTful API is configured		
Summary	<ul style="list-style-type: none"> - Client Application [APP1] has two instances one is in vehicle ECU [ECU1] and another is in backend [ECU2]. While Server Application [APP2] is in vehicle ECU [ECU1] only. - Request is sent and response is received from server. - Server application [APP2] provides a service [SERVICE2] with resource [Resource1] service [SERVICE2] is requested by [APP1] by HTTP and inter Process Communication (IPC). - Response from server is processed and then client unsubscribe from the event. - Client is stopped. Note: In-vehicle ECU instance of [APP1] uses IPC to request the service while instance of [APP1] in backend request [SERVICE2] using HTTP. 		
Pre-conditions	<ul style="list-style-type: none"> - [REST Tester] is connected to ECU. - ECU is in Machine State Parking. 		





Post-conditions	TCP connections between [REST Tester] and both ECUs are closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[REST Client Application] [APP1] [ECU1] Send Request (using IPC) to get status of Resource1	
Step 2	[REST Server Application] [APP2] [ECU1]	Positive response is received from Server.
Step 3	[REST Client Application] [APP1] [ECU2] Send Request (using HTTP) to get status of Resource1	
Step 4	[REST Server Application] [APP2] [ECU1]	Positive response is received from Server

4.3.4 [STS_REST_00004] Data Representation

Test Objective	To verify the Abstraction of the used payload format (e.g. JSON or XML).		
ID	STS_REST_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	REST	State	Draft
Trace to RS Criteria	RS_CM_00301, RS_CM_00305, RS_CM_00306, RS_CM_00308, RS_CM_00307, RS_CM_00313		
Reference to Test Environment	STC_REST_00002		
Configuration Parameters	RESTful API is configured		
Summary	<ul style="list-style-type: none"> - Client and Server Applications communicates as per RESTful communication. - First client is set up and request is created with URI and Methods (GET/PUT/ POST/DELETE/OPTIONS). - Request is sent and response is received from server as Object Graph having payload format JSON or XML. - Response from server is processed and then client unsubscribe from the event. - Client is stopped. 		
Pre-conditions	<ul style="list-style-type: none"> - [REST Tester] is connected to ECU. - ECU is in Machine State Parking. 		
Post-conditions	TCP connections between [REST Tester] and both ECUs are closed.		
Main Test Execution			
Test Steps			Pass Criteria
Step 1	[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name>		





	<p>ContentLength: <length> △</p> <p>Accept: <application/json></p> <p>Version: HTTP/1.1</p>	
Step 2	<p>[REST Server Application]</p> <p>Server Response: HTTP/1.1 200 OK</p> <p>Content-Type: <application/json></p> <p>Status: <Status1></p> <p>URI : http://<host-name>:<port>/SERVICE1//Resource1/Element1/<Status></p>	Positive response is received from Server.
Step 3	<p>[REST Client Application]</p> <p>Send Request to get status of Resource1/Element1</p> <p>Method: GET</p> <p>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status</p> <p>Host: <host-name></p> <p>ContentLength: <length></p> <p>Accept: <application/xml></p> <p>Version: HTTP/1.1</p>	
Step 4	<p>[REST Server Application]</p> <p>Server Response: HTTP/1.1 200 OK</p> <p>Content-Type: <application/xml></p> <p>Status: <Status1></p> <p>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status></p> <p><resources></p> <p><resource>Resource1</resource></p> <p><elements></p> <p><status>Status1</status></p> <p></elements></p> <p><elements></p> <p><status>Status2</status></p> <p></elements></p> <p></resources></p>	Positive response is received from Server.
Step 5	<p>[REST Client Application]</p> <p>Send Request to get status of Resource1/Element1</p> <p>Method: GET</p> <p>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status</p> <p>Host: <host-name></p> <p>ContentLength: <length></p> <p>Accept: <application/json></p> <p>Version: HTTP/1.1</p>	
Step 6	<p>[REST Server Application]</p> <p>Server Response: HTTP/1.1</p> <p>Content-Type: <application/xml></p> <p>Status: <Status1></p> <p>URI : http://<host-name>:<port>/SERVICE1//Resource1/Element1/<Status></p>	Response is rejected due to mismatch in Content type.



Step 7	[REST Client Application] Send Request to get status of Resource1/Element1 Method: GET URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status Host: <host-name> ContentLength: <length> Accept: <application/xml> Version: HTTP/1.1	
Step 8	[REST Server Application] Server Response: HTTP/1.1 Content-Type: <application/json> Status: <Status1> URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status>	Response is rejected due to mismatch in Content type.

4.3.5 [STS_REST_00005] Event communication with Web-sockets

Test Objective	To verify the event-based communication with the WebSocket protocol.		
ID	STS_REST_00005	AUTOSAR Releases	R18-10
Affected Functional Cluster	REST	State	Draft
Trace to RS Criteria	RS_CM_00314		
Reference to Test Environment	STC_REST_00001		
Configuration Parameters	RESTful API is configured		
Summary	<ul style="list-style-type: none"> - Client sends a handshake request to server to establish WebSocket connection. - Server returns a WebSocket handshake response. - Once the connection is established both client and server can listen for events. - Event subscription message is sent as JSON over WebSocket channel. - Then Event cancellation message is sent as JSON over WebSocket channel - Response from server is processed and then client unsubscribe from the event. - Client is stopped. 		
Pre-conditions	<ul style="list-style-type: none"> - [REST Tester] is connected to ECU. - ECU is in Machine State Parking. 		
Post-conditions	TCP connections between [REST Tester] and both ECUs are closed.		
Main Test Execution			
Test Steps			Pass Criteria





Step 1	<p>[REST Client Application] Send handshake request to server to establish WebSocket connection.</p> <p>GET /<protocol> HTTP/1.1</p> <p>Host: server.<URL></p> <p>Upgrade: websocket</p> <p>Connection: Upgrade</p> <p>Sec-WebSocket-Key: <key>==</p> <p>Origin: http://<URL></p> <p>Sec-WebSocket-Protocol: <protocol></p> <p>Sec-WebSocket-Version: <version></p>	
Step 2	<p>[REST Server Application]</p> <p>Handshake from the server:</p> <p>HTTP/1.1 101 Switching Protocols</p> <p>Upgrade: websocket</p> <p>Connection: Upgrade</p> <p>Sec-WebSocket-Accept: <key>0=</p> <p>Sec-WebSocket-Protocol: <protocol></p>	<p>Positive Handshake Response is received from Server.</p>
Step 3	<p>WebSocket channel is opened during the first Event subscription and subscription message is sent as JSON over WebSocket channel.</p> <p>"type": "subscribe"</p>	
Step 4	<p>[REST Server Application]</p> <p>Server Responses to subscription message</p>	<p>Positive Subscription Response is received from Server.</p>
Step 5	<p>[REST Client Application] Event cancellation message is sent as JSON over WebSocket channel</p> <p>"type": "unsubscribe"</p>	
Step 6	<p>[REST Server Application]</p> <p>Server Responses to cancellation message</p>	<p>Positive cancellation Response is received from Server.</p>
Step 7	<p>[REST Client Application] Request Error message from server.</p>	
Step 8	<p>[REST Server Application]</p> <p>Server sends the Event error messages as JSON over the WebSocket channel.</p> <p>"type": "error"</p>	<p>Error message is received from Server.</p>

5 Test configuration and test steps for Execution Management

5.1 Test System

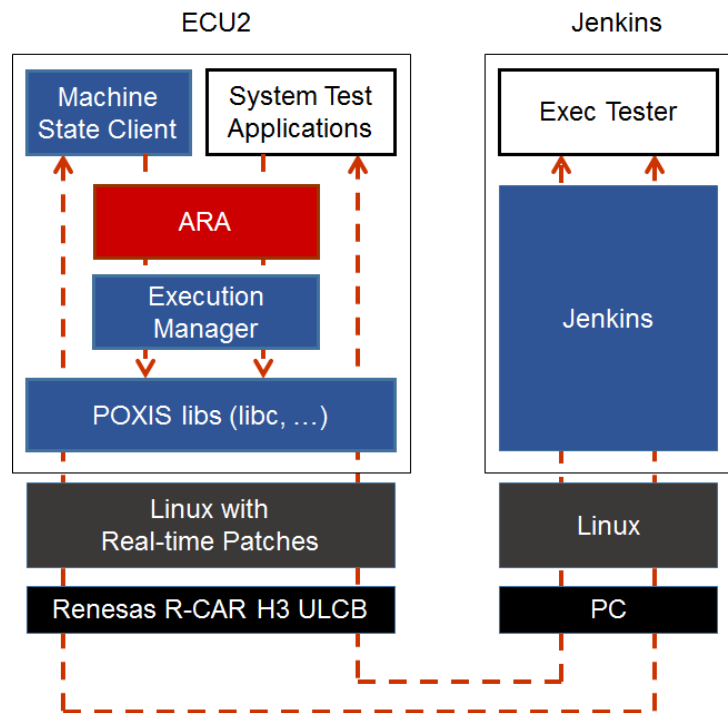


Figure 5.1: Illustration of test setup for Execution Management.

5.1.1 Test configurations

5.1.1.1 STC_EMO_00001

Configuration ID	STC_EMO_00001
Description	Standard Jenkins server for Execution Management test
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Execution Management test (Exec Tester) is connected via Ethernet to ECU2 hosting the System Test Applications [APP2], [APP3], [APP4] and [APP5].

The Exec Tester is supposed to check the pass criteria.

The communication between Exec Tester and the applications on the ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

5.1.1.1.1 Machine Manifest

Machine States	<i>Startup (Initial Mode)</i>
	<i>Shutdown</i>
	<i>Restart</i>
	<i>Driving</i>
	<i>Parking</i>

5.1.1.1.2 Application Manifest

Application Name	APP2		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP3		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP4		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP5		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>

5.1.1.2 STC_EMO_00002

Configuration ID	STC_EMO_00002
Description	Standard Jenkins server for Execution Management test
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Execution Management test (Exec Tester) is connected via Ethernet to ECU2 hosting the System Test Applications [APP2], [APP3], [APP4], [APP5] and [APP6].

The Exec Tester is supposed to check the pass criteria.

The communication between Exec Tester and the applications on the ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

5.1.1.2.1 Machine Manifest

Machine States	<i>Startup (Initial Mode)</i>
	<i>Shutdown</i>
	<i>Restart</i>
	<i>Driving</i>
	<i>Parking</i>
Function Groups	
FG1	<i>Off</i>
	<i>Running</i>
	<i>Fallback</i>
	<i>Diag</i>
FG2	<i>Off</i>
	<i>On</i>
	<i>Activate</i>

5.1.1.2.2 Application Manifest

Application Name	APP2		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP3		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
		executionDependency	[APP2]. <i>Running</i>
Application Name	APP4		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
		executionDependency	[APP3]. <i>Running</i>
Application Name	APP5		
Process	ModeDependentStartupConfig	functionGroup	[FG2]. <i>On</i> and [FG2]. <i>Activate</i>
Application Name	APP6		
Process	ModeDependentStartupConfig	functionGroup	[FG2]. <i>Activate</i>

5.1.1.3 STC_EMO_00003

Configuration ID	STC_EMO_00003
Description	Standard Jenkins server for Execution Management test
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Execution Management test (Exec Tester) is connected via Ethernet to ECU2 hosting the System Test Applications [APP2], [APP3], [APP4] and [APP5].

The Exec Tester is supposed to check the pass criteria.

The communication between Exec Tester and the applications on the ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

5.1.1.3.1 Machine Manifest

Machine States	<i>Startup (Initial Mode)</i>
	<i>Shutdown</i>
	<i>Restart</i>
	<i>Driving</i>
	<i>Parking</i>

5.1.1.3.2 Application Manifest

Application Name	APP2		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP3		
Process	ModeDependentStartupConfig	machineMode	<i>Driving</i>
Application Name	APP4		
Process	ModeDependentStartupConfig	machineMode	<i>Parking</i>
Application Name	APP5		
Process	ModeDependentStartupConfig	machineMode	<i>Parking</i>

5.2 Test cases

5.2.1 [STS_EMO_00001] Startup of applications with change of machine state.

Test Objective	Verification, that the execution management functional cluster can perform a change of Machine State and that applications associated with the new Machine State are started.		
ID	STS_EMO_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Execution Management	State	Draft
Trace to RS Criteria	RS_EM_00100, RS_EM_00101		
Reference to Test Environment	STC_EMO_00001		
Configuration Parameters	<ul style="list-style-type: none"> Machine State Driving, in which all System Test Applications [APP2], [APP3], [APP4] and [APP5] shall start is defined. 		





Summary	<p>When initialized the system state is <i>Startup</i>.</p> <p>A change of Machine State from <i>Startup</i> to <i>Parking</i> is requested and it is verified that [APP2], [APP3], [APP4] and [APP5] are not started.</p> <p>A change of Machine State from <i>Parking</i> to <i>Driving</i> is requested and the startup of the applications [APP2], [APP3], [APP4] and [APP5] associated with this Machine State is verified.</p>	
Pre-conditions	<ul style="list-style-type: none"> - Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - ECU2 is in Machine State <i>Startup</i>. - Operating system on ECU2 has booted. 	
Post-conditions	TCP connection between Exec Tester and ECU2 is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[Exec Tester] Request change of Machine State to <i>Parking</i> for ECU2.	
Step 2	[SM] Request for change of Machine State to <i>Parking</i> from Execution Manager.	Machine State for ECU2 is changed to <i>Parking</i> .
Step 3	[Exec Tester] Query execution status of [APP2].	[APP2] is not executed.
Step 4	[Exec Tester] Query execution status of [APP3].	[APP3] is not executed.
Step 5	[Exec Tester] Query execution status of [APP4].	[APP4] is not executed.
Step 6	[Exec Tester] Query execution status of [APP5].	[APP5] is not executed.
Step 7	[Exec Tester] Request change of Machine State to <i>Driving</i> for ECU2.	
Step 8	[SM] Request for change of Machine State to <i>Driving</i> from Execution Manager.	Machine State for ECU2 is changed to <i>Driving</i> .
Step 9	[Exec Tester] Query execution status of [APP2].	[APP2] is executed.
Step 10	[Exec Tester] Query execution status of [APP3].	[APP3] is executed.
Step 11	[Exec Tester] Query execution status of [APP4].	[APP4] is executed.
Step 12	[Exec Tester] Query execution status of [APP5].	[APP5] is executed.

5.2.2 [STS_EMO_00002] Shutdown of applications with change of machine state to Shutdown

Test Objective	Verification, that the execution management functional cluster executes a well-defined shutdown sequence for all configured and running applications, When shut-down is initiated		
ID	STS_EMO_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Execution Management	State	Draft
Trace to RS Criteria	RS_EM_00100, RS_EM_00101		
Reference to Test Environment	STC_EMO_00001		
Configuration Parameters	<ul style="list-style-type: none"> - Machine State Driving, in which all System Test Applications [APP2], [APP3], [APP4] and [APP5] shall start is defined. - ECU ID for ECU2 is set to ECU2 - [APP2] has LT Application ID APPID2. - Context ID for [APP2] is set to CTX2 - [APP3] has LT Application ID APPID3. - Context ID for [APP3] is set to CTX3 - [APP4] has LT Application ID APPID4. - Context ID for [APP4] is set to CTX4 - [APP5] has LT Application ID APPID5. - Context ID for [APP5] is set to CTX5 		
Summary	A change of Machine State from <i>Driving</i> to <i>Shutdown</i> is requested and the Shutdown of the applications [APP2], [APP3], [APP4] and [APP5] is verified by logging the messages at the termination of application.		
Pre-conditions	<ul style="list-style-type: none"> - Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - ECU2 is in Machine State Driving. - Operating system on ECU2 has booted. - Applications [APP2], [APP3], [APP4] and [APP5] are registered for logging and default log level is set to Verbose. 		
Post-conditions	TCP connection between Exec Tester and ECU2 is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[Exec Tester] Request change of Machine State to <i>Shutdown</i> for ECU2.		
Step 2	[SM] Request for change of Machine State to <i>Shutdown</i> from Execution Manager.		Machine State for ECU2 is changed to <i>Shutdown</i> .
Step 3	[Exec Tester] Observe the log for applications [APP2], [APP3], [APP4] and [APP5]		Message with context ID CTX2 and application ID APPID2 is received which is logged at [APP2] application termination Message with context ID CTX3 and application ID APPID3 is received which is logged at [APP3] application termination ▾





		△ Message with context ID CTX4 and application ID APPID4 is received which is logged at [APP4] application termination Message with context ID CTX5 and application ID APPID5 is received which is logged at [APP5] application termination
--	--	---

5.2.3 [STS_EMO_00003] Ordered Startup and Shutdown of Executables based on the dependency with other processes

Test Objective	Verification, that the execution management functional cluster can perform a change of Machine State and that applications associated with the new Machine State are started considering the dependency with other processes. Also to verify the ordered shutdown of the processes.		
ID	STS_EMO_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Execution Management	State	Draft
Trace to RS Criteria	RS_EM_00100, RS_EM_00101, RS_EM_00103		
Reference to Test Environment	STC_EMO_00002		
Configuration Parameters	<ul style="list-style-type: none"> - Machine State <i>Driving</i>, in which System Test Applications [APP2], [APP3] and [APP4] shall start is defined. Dependency with other process is configured as mentioned in section 5.2.1.2.2 Application Manifest. - ECU ID for ECU2 is set to <i>ECU2</i> - [APP2] has LT Application ID <i>APPID2</i> - Context ID for [APP2] is set to <i>CTX2</i> - [APP3] has LT Application ID <i>APPID3</i> - Context ID for [APP3] is set to <i>CTX3</i> - [APP4] has LT Application ID <i>APPID4</i> - Context ID for [APP4] is set to <i>CTX4</i> - [APP5] has LT Application ID <i>APPID5</i> - Context ID for [APP5] is set to <i>CTX5</i> - [APP6] has LT Application ID <i>APPID6</i> - Context ID for [APP6] is set to <i>CTX6</i> 		
Summary	<p>When initialized the system state is <i>Startup</i>.</p> <p>A change of Machine State from <i>Startup</i> to <i>Driving</i> is requested and the startup of the applications [APP2], [APP3] and [APP4] associated with this Machine State are verified in the order of [APP2], [APP3] and [APP4] by logging the messages at the Start of application processes.</p> <p>A change of Machine State from <i>Driving</i> to <i>Parking</i> is requested and the termination of the applications [APP2], [APP3] and [APP4] is verified in the order of [APP4], [APP3] and [APP2] by logging the messages at the termination of application processes.</p>		





Pre-conditions	<ul style="list-style-type: none"> - Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - ECU2 is in Machine State <i>Startup</i>. - Function Group State for [FG2] is <i>Off</i>. - Operating system on ECU2 has booted. 	
Post-conditions	TCP connection between Exec Tester and ECU2 is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[Exec Tester] Request change of Machine State to <i>Driving</i> for ECU2.	
Step 2	[SM] Request for change of Machine State to <i>Driving</i> from Execution Manager.	Machine State for ECU2 is changed to <i>Driving</i> .
Step 3	[Exec Tester] Observe the log for applications [APP2]	Message with context ID <i>CTX2</i> and application ID <i>APPID2</i> is received which is logged at [APP2] application startup
Step 4	[Exec Tester] Observe the log for applications [APP3]	Message with context ID <i>CTX3</i> and application ID <i>APPID3</i> is received which is logged at [APP3] application startup
Step 5	[Exec Tester] Observe the log for applications [APP4]	Message with context ID <i>CTX4</i> and application ID <i>APPID4</i> is received which is logged at [APP4] application startup
Step 6	[Exec Tester] Request change of Machine State to <i>Shutdown</i> for ECU2.	
Step 7	[SM] Request for change of Machine State to <i>Parking</i> from Execution Manager.	Machine State for ECU2 is changed to <i>Parking</i> .
Step 8	[Exec Tester] Observe the log for applications [APP4]	Message with context ID <i>CTX4</i> and application ID <i>APPID4</i> is received which is logged at [APP4] application termination
Step 9	[Exec Tester] Observe the log for applications [APP3]	Message with context ID <i>CTX3</i> and application ID <i>APPID3</i> is received which is logged at [APP3] application termination
Step 10	[Exec Tester] Observe the log for applications [APP2]	Message with context ID <i>CTX2</i> and application ID <i>APPID2</i> is received which is logged at [APP2] application termination

5.2.4 [STS_EMO_00004] Startup of applications with change of Function Group state

Test Objective	Verification, that the execution management functional cluster can perform a change of Function Group State and that Applications associated with the new Function Group State are started.		
ID	STS_EMO_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	Execution Management	State	Draft
Trace to RS Criteria	RS_EM_00100, RS_EM_00101		
Reference to Test Environment	STC_EMO_00002		
Configuration Parameters	<ul style="list-style-type: none"> - Function Group State <i>Activate</i> and Function Group State <i>On</i> of [FG2] in which System Test Application [APP5] shall start is defined. - Function Group State <i>Activate</i> of [FG2] in which System Test Application [APP6] shall start is defined 		
Summary	<p>When initialized the Function Group State of [FG2] is <i>Off</i>.</p> <p>A change of Function Group State of [FG2] to <i>On</i> is requested and the startup of the application [APP5] associated with this Function Group State is verified.</p> <p>A change of Function Group State of [FG2] to <i>Activate</i> is requested and the startup of [APP6] associated with this Function Group State is verified.</p>		
Pre-conditions	<ul style="list-style-type: none"> - Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - Function Group State [FG2] is <i>Off</i>. - Operating system on ECU2 has booted. 		
Post-conditions	TCP connection between Exec Tester and ECU2 is closed.		
Main Test Execution			
Test Steps			Pass Criteria
Step 1	[Exec Tester] Request change of Function Group State [FG2] to <i>On</i> .		
Step 2	[SM] Request for change of Function Group State [FG2] to <i>On</i> from Execution Manager.		Function Group State [FG2] for ECU2 is changed to <i>On</i> .
Step 3	[Exec Tester] Query execution status of [APP5].		[APP5] is executed.
Step 4	[Exec Tester] Request change of Function Group State [FG2] to <i>Activate</i> .		
Step 5	[SM] Request for change of Function Group State [FG2] to <i>Activate</i> from Execution Manager.		Function Group State [FG2] for ECU2 is changed to <i>Activate</i> .
Step 6	[Exec Tester] Query execution status of [APP6].		[APP6] is executed.

5.2.5 [STS_EMO_00005] Execution Management shall prevent Processes from directly starting other Processes

Test Objective	Verification that the execution management shall prevent Processes from directly starting other Processes		
ID	STS_EMO_00005	AUTOSAR Releases	R18-10
Affected Functional Cluster	Execution Management	State	Draft
Trace to RS Criteria	RS_EM_00009		
Reference to Test Environment	STC_EMO_00003		
Configuration Parameters	<ul style="list-style-type: none"> - Machine State Driving, in which all System Test Applications [APP2] and [APP3] shall start is defined and Machine State Parking in which Applications [APP4] and [APP5] shall start is defined. - Each of the Applications [APP2], [APP3], [APP4] and [APP5] have one Executable invoked by a Process 		
Summary	<p>A change of Machine State from <i>Startup</i> to <i>Driving</i> is requested. Start of [APP2] and [APP3] Processes from Execution Manager is checked.</p> <p>Create or fork a Process from [APP2] Process and verify that no child Processes are created from [APP2] Process.</p> <p>Execute [APP5] Process from [APP3] Process and verify that the [APP5] Process is not invoked from [APP3] Process.</p>		
Pre-conditions	<ul style="list-style-type: none"> - Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - ECU2 is in Machine State <i>Startup</i>. - Operating system on ECU2 has booted. 		
Post-conditions	TCP connection between Exec Tester and ECU2 is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[Exec Tester] Request change of Machine State to Driving for ECU2.		
Step 2	[SM] Request for change of Machine State to <i>Driving</i> from Execution Manager.		Machine State for ECU2 is changed to <i>Driving</i> .
Step 3	Query execution status of [APP2]		[APP2] Process is executed
Step 4	[APP2] Fork or create a Process from [APP2]		
Step 5	[Exec Tester] Get the Process ID of the Execution Manager		Received the Process ID of Execution Manager. <i>EXMPID(1)</i>
Step 6	[Exec Tester] Get the Process ID of [APP2] Process		Received the Process ID of [APP2] Process <i>APP2PID</i>
Step 7	[Exec Tester] Get the Parent Process ID of [APP2] Process		The Parent Process ID of [APP2] Process is received as <i>EXMPID(1)</i>
Step 8	[Exec Tester] Get the Child Processes of Process ID <i>APP2PID</i>		No child Processes of [APP2] Process shall be received.
Step 9	Query execution status of [APP3]		[APP3] Process is executed





Step 10	[APP3] Execute or Invoke [APP5] Process from [APP3] Process	[APP5] Process is not executed
----------------	--	--------------------------------

6 Test configuration and test steps for Diagnostics

6.1 Test System

6.1.1 Test configurations

Configuration ID	STC_DIAG_00001
Description	Standard Jenkins server for diagnostic test
ECU 1	Intel Minnowboard Turbot, 192.168.100.5
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server running the job with the [Diagnostic Tester] is connected via Ethernet to [ECU1] hosting the System Test Application [APP1] respectively. The [Diagnostic Tester] will open TCP connections on port 13400 and send diagnostic data as UDS requests in DoIP packets.

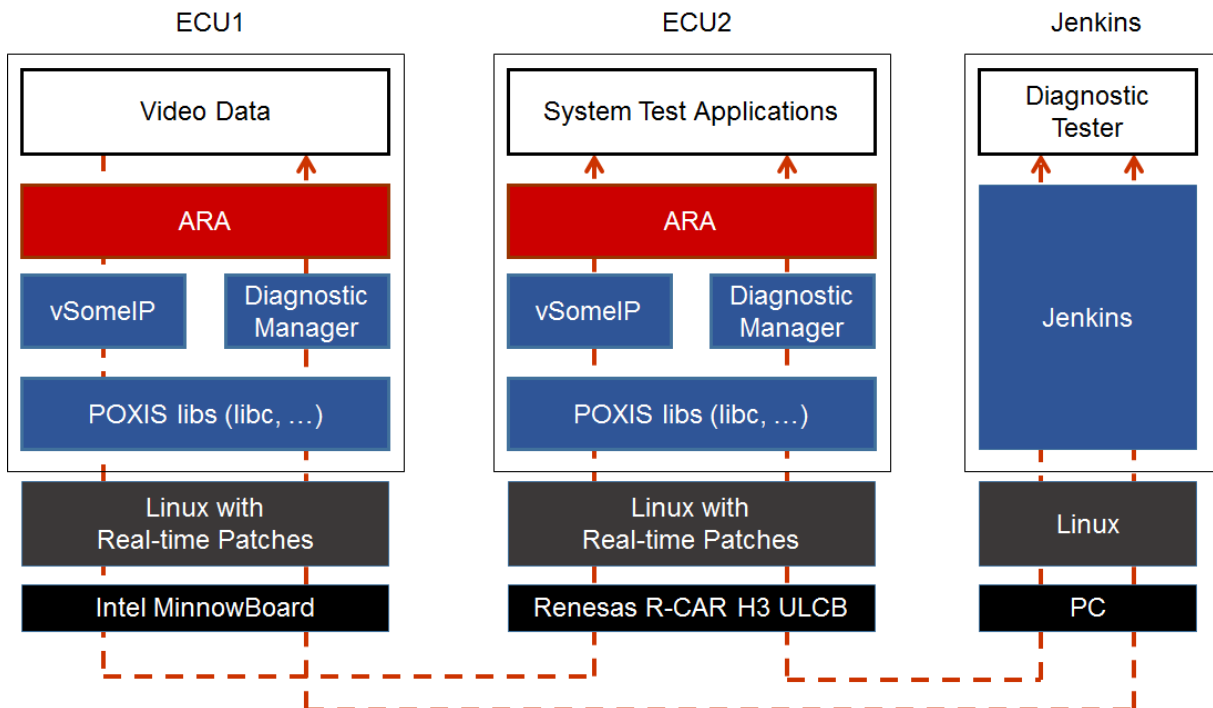


Figure 6.1: Illustration of test setup for Diagnostics.

6.2 Test cases

6.2.1 [STS_DIAG_00001] Utilization of Diagnostic service ReadDataByIdentifier (0x22) by external Tester via UDS messages over DoIP.

Test Objective	Verification of correct behavior of Diagnostic service ReadDataByIdentifier (0x22) by external Tester via UDS messages over DoIP.		
ID	STS_DIAG_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Diagnostic	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	- Diagnostics module: <ul style="list-style-type: none"> • Service instance for service ReadDataByIdentifier with DID <0x0001> is configured. • Service instance with DID <0x0099> is NOT configured. 		
Summary	This basic test tries to query the value of a variable contained by [APP1] on [ECU1] over the AP Diagnostics Module. The UDS service ReadDataByIdentifier (0x22) is used. The AP Diagnostics Module has to call a service in the Application Layer to retrieve the requested information and send it back as UDS response. If an unknown identifier is queried, a negative response must be sent.		
Pre-conditions	- [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port. - Software components on [ECU1] are initialized.		
Post-conditions	TCP connection between [Diagnostic Tester] and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[Diagnostic Tester] Send UDS Request to query value of <int1>: UDS-Service: ReadDataByIdentifier UDS-Payload: 0x22 ...		
Step 2	[APP1] Start mechanism to read the value of <int1>.		
Step 3	[Diagnostic Tester] Receive UDS response and save value of <int1> in <var1>.		Positive response received (0x62 ...). Payload of UDS response contains DID data with value of <int1>.
Step 4	[APP1] Start mechanism to change the value of <int1> by <delta>.		
Step 5	[Diagnostic Tester] Send UDS Request to query value of <int1>: UDS-Service: ReadDataByIdentifier UDS-Payload: 0x22 ...		
Step 6	[APP1] Start mechanism to read value of <int1> and return it as DID data.		





Step 7	[Diagnostic Tester] Receive UDS response and save value of <int1> in <var2>.	Positive response received (0x62 ...). Payload of UDS response contains DID data. Compare values of <var1> and <var2>. <var2> should be greater than <var1> by <delta> i.e. <var2>=<var1> + <delta>.
Step 8	[Diagnostic Tester] Send UDS Request to query data with a non-implemented DID: UDS-Service: ReadDataByIdentifier UDS-Payload: 0x22 ...	Tester receives negative response: 0x7F 0x22 0x31.

6.2.2 [STS_DIAG_00002] Utilization of Diagnostic service RoutineControl (0x31) by external Tester via UDS messages over DoIP.

Test Objective	Verification of correct behavior of Diagnostic service RoutineControl (0x31) by external Tester via UDS messages over DoIP.		
ID	STS_DIAG_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Diagnostic	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	<p>- The following service is configured [SERVICE1] in [APP1] - In this [SERVICE1], two different contents are available</p> <ul style="list-style-type: none"> • <Content1> • <Content2> <p>- Diagnostics module:</p> <ul style="list-style-type: none"> • Service instance for service RoutineControl with RID <0x0001> is configured and only available in Extended Diagnostic Session. • Service Diagnostic Session Control is configured. 		
Summary	This test tries to start a routine in [APP1] over the AP Diagnostics Module and the UDS service RoutineControl (0x31). In DefaultSession, execution is not allowed and a negative response is sent. After switching to ExtendedDiagnosticSession, the routine is started and a positive response is sent.		
Pre-conditions	<p>- [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port.</p> <p>- Software components on [ECU1] are initialized.</p> <p>- [APP1] sends <Content1> via [SERVICE1].</p>		
Post-conditions	TCP connection between Jenkins server and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[Diagnostic Tester] Send UDS request to change content of [SERVICE1]: UDS-Service: RoutineControl UDS-Payload: 0x31 0x01 ...	Negative response received: Service Not Supported in Active Session (0x7F 0x31 0x7F).	





Step 2	[Diagnostic Tester] Send UDS request to start an Extended Diagnostic Session: UDS-Service: DiagnosticSessionControl UDS-Payload: 0x10 0x03	Positive response received (0x50 0x03).
Step 3	[Diagnostic Tester] Send UDS request to change content of [SERVICE1] from <Content1> to <Content2>: UDS-Service: RoutineControl UDS-Payload: 0x31 0x01 ...	
Step 4	[APP1] Start mechanism to change content of [SERVICE1] from <Content1> to <Content2>	Content of Service is changed to <Content2>
Step 5	[APP1] Return from Subfunction Start of Routine with RID <0x0001>.	
Step 6	[Diagnostic Tester] Receive UDS response.	Positive response received (0x71 ...).

6.2.3 [STS_DIAG_00003] Utilization of Diagnostic service TesterPresent (0x3E) by External Tester via UDS messages over DoIP.

Test Objective	Verification of correct behavior of Diagnostic service TesterPresent (0x3E) by External Tester via UDS messages over DoIP.		
ID	STS_DIAG_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Diagnostic	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	Diagnostics module: <ul style="list-style-type: none"> • Service instance for service RoutineControl with RID <0x0001> is configured and only available in Extended Diagnostic Session. • Service Diagnostic Session Control and Extended Diagnostic Session time out is configured. • TesterPresent is configured. 		
Summary	TesterPresent request is sent to indicate that previously activated non-default (e.g. extended) session will still be active. The UDS service RoutineControl (0x31) is executed to check if Extended session is active (Any other service which is supported in extended session may be used). Positive response is received for the TesterPresent request if suppressPosRspMsgIndicationBit is set to FALSE. No response is expected (by Client) from Server if, suppressPosRspMsgIndicationBit is set to TRUE		
Pre-conditions	- [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port. - Software components on [ECU1] are initialized.		
Post-conditions	TCP connection between Jenkins server and [ECU1] is closed.		
Main Test Execution			
Test Steps			Pass Criteria





Step 1	[Diagnostic Tester] Send UDS request to start an Extended Diagnostic Session: UDS-Service: DiagnosticSessionControl(SID 0x10) UDS-Payload: 0x10 0x03	Positive response received (0x50 0x03).
Step 2	[Diagnostic Tester] Wait for time <t1> such that <t1> is less than Diagnostic session timer timeout.	
Step 3	[Diagnostic Tester] Send UDS request Tester Present with suppressPosRspMsg IndicationBit is set to FALSE. UDS-Service: TesterPresent (SID 0x3E) UDS-Payload: 0x3E 0x00	Positive response received (0x7E 0x00).
Step 4	[Diagnostic Tester] Wait for time <t2> such that - 1) <t2> is greater than Diagnostic session timer timeout. 2) <t2> is less than sum of Extended session timer and Diagnostic session timer timeout.	
Step 5	[Diagnostic Tester] Send UDS request RoutineControl to confirm if Extended Session is active. UDS-Service: RoutineControl (SID 0x31) UDS-Payload: 0x31 0x01 ...	Positive response received (0x71 ...).
Step 6	[Diagnostic Tester] Stop sending TesterPresent and wait for Extended Diagnostic Session to time out	
Step 7	[Diagnostic Tester] Send UDS request RoutineControl to confirm if Extended Session is active. UDS-Service: RoutineControl UDS-Payload: 0x31 0x01 ...	Negative response received: Service Not Supported in Active Session (0x7F 0x31 0x7F (NRC)).
Step 8	[Diagnostic Tester] Send UDS request to start an Extended Diagnostic Session: UDS-Service: DiagnosticSessionControl UDS-Payload: 0x10 0x03	Positive response received (0x50 0x03).
Step 9	[Diagnostic Tester] Wait for time <t1> such that <t1> is less than Diagnostic session timer timeout.	
Step 10	[Diagnostic Tester] Send UDS request TesterPresent with suppressPosRspMsg IndicationBit is set to TRUE. UDS-Service: TesterPresent UDS-Payload: 0x3E 0x80	No response received for UDS request TesterPresent.





Step 11	[Diagnostic Tester] Wait for time <t2> such that - 1) <t2> is greater than Diagnostic session timer timeout. 2) <t2> is less than sum of Extended session timer and Diagnostic session timer timeout.	
Step 12	[Diagnostic Tester] Send UDS request RoutineControl to confirm if Extended Session is active. UDS-Service: RoutineControl UDS-Payload: 0x31 ...	Positive response received (0x71 ...).

6.2.4 [STS_DIAG_00004] Utilization of Diagnostic service WriteDataByIdentifier (0x2E) by External Tester via UDS messages over DoIP.

Test Objective	Verification of correct behavior of Diagnostic service WriteDataByIdentifier (0x2E) by External Tester via UDS messages over DoIP.		
ID	STS_DIAG_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	Diagnostic	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	Diagnostics module: - Service instances for service ReadDataByIdentifier and WriteDataByIdentifier with DID <0x0001> are configured.		
Summary	This basic test tries to query the value of <int1> contained by [APP1] on [ECU1] over the AP Diagnostics Module. The UDS service ReadDataByIdentifier (0x22) is used and then the value of <int1> is overwritten by UDS service WriteDataByIdentifier (0x2E). Overwritten value of the variable <int1> is read back using UDS service ReadDataByIdentifier (0x22).		
Pre-conditions	- [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port - Software components on [ECU1] are initialized.		
Post-conditions	TCP connection between [Diagnostic Tester] and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[Diagnostic Tester] Send UDS Request to query value of <int1>: UDS-Service: ReadDataByIdentifier UDS-Payload: 0x22 ...		
Step 2	[APP1] Wait for invocation.	Implementation of method Read for DID <0x0001> is invoked.	
Step 3	[Diagnostic Tester] Receive UDS response with value of <int1>.	Positive response received (0x62 ...). Payload of UDS response contains DID data with value of <int1>.	





Step 4	[Diagnostic Tester] Send UDS Request to overwrite value of <int1> with <int2> UDS-Service: WriteDataByIdentifier UDS-Payload: 0x2E ...	
Step 5	[Diagnostic Tester] Receive UDS response.	Positive response received (0x6E ...) after successful write.
Step 6	[Diagnostic Tester] Send UDS request to query value of <int1> UDS-Service: ReadDataByIdentifier UDS-Payload: 0x22 ...	
Step 7	[APP1] Wait for invocation.	Implementation of method Read for DID <0x0001> is invoked.
Step 8	[Diagnostic Tester] Receive UDS response with value of <int1> and store it in <var>.	Positive response received (0x62 ...). Payload of UDS response contains DID data with value of <int1>.
Step 9	[Diagnostic Tester] Compare <var> and <int2> values.	Both values should be equal.

7 Test configuration and test steps for Logging and Tracing

7.1 Test System

7.1.1 Test configurations

Configuration ID	STC_LT_00001
Description	Standard Jenkins server for LT test
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the LT Tester, is connected via Ethernet to [ECU1] hosting the System Test Application [APP1] and [ECU2] hosting the System Test Application [APP2]. The LT Tester opens TCP connections on port 3490 and receives log messages from the LT module.

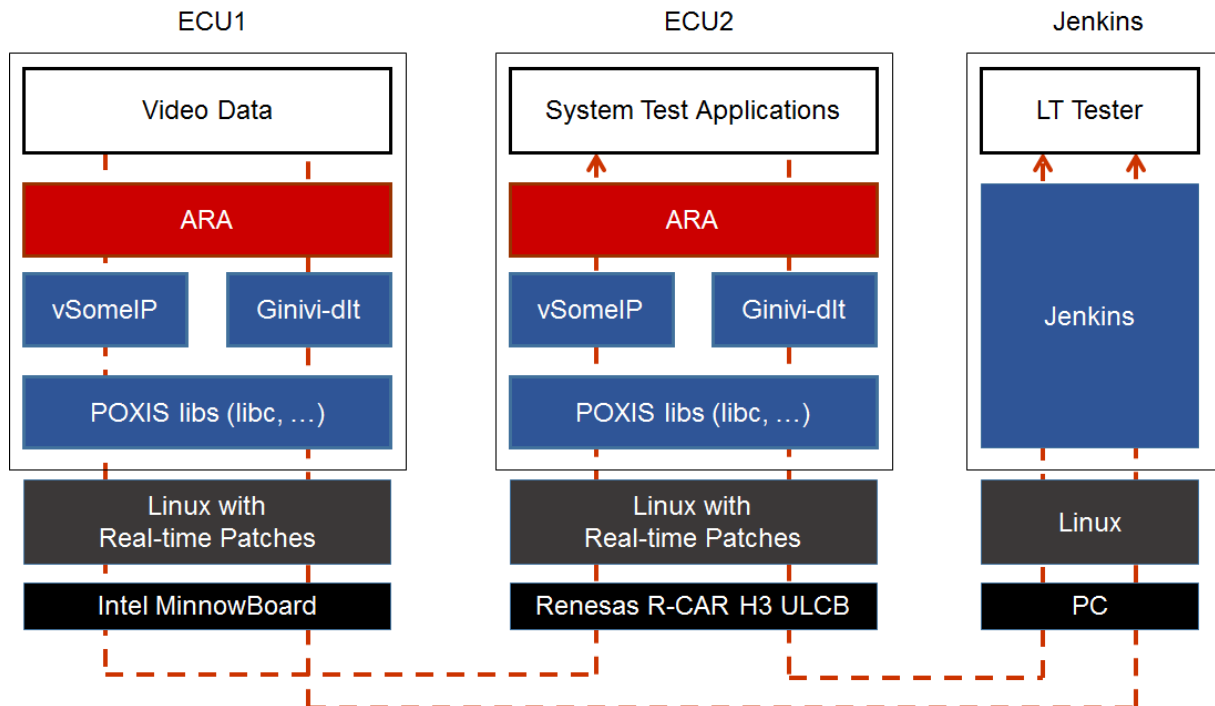


Figure 7.1: Illustration of test setup for Logging and Tracing.

7.2 Test cases

7.2.1 [STS_LT_00001] Receiving of log messages from LT module by external Tester and remote control of application's default log level.

Test Objective	Verification that all sent log messages from LT module are received by external Tester, that they carry the correct attributes like Application ID and ECU ID, and that the remote control of the application's default log level works.		
ID	STS_LT_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Logging and Tracing	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_LT_00001		
Configuration Parameters	<ul style="list-style-type: none"> - LT module in ECU1 is configured properly: - ECU ID for ECU1 is set to ECU1 - [APP1] has LT Application ID APPID1. - Context ID for [APP1] is set to CTX1 		
Summary	The LT Tester has to connect to the LT module, which has to receive and forward the log messages from the Application Layer. First, log messages on all log levels with correct attributes are expected. Then the applications default log level is consecutively lowered to more restrictive values and it is checked, whether the respective log messages disappear.		
Pre-conditions	[LT Tester] is connected to [ECU1] via TCP socket on Port 3490. <ul style="list-style-type: none"> • Software components on [ECU1] are initialized. • Video Provider's default log level is set to Verbose. 		
Post-conditions	TCP connection between [LT Tester] and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[LT Tester] Receive log messages.		Tester receives log messages every 0.5 seconds. The messages are received for all log levels in context with ID CTX1 and contain ECU ID ECU1, and Application ID APPID1.
Step 2	[LT Tester] Send request to query change of [APP1] default log level to Debug.		Messages with log level Verbose are no longer received. Messages with lower log level are still coming in.
Step 3	[LT Tester] Send request to query change of [APP1] default log level to Info.		Messages with log level Debug are no longer received. Messages with lower log level are still coming in.
Step 4	[LT Tester] Send request to query change of [APP1] default log level to Warn.		Messages with log level Info are no longer received. Messages with lower log level are still coming in.
Step 5	[LT Tester] Send request to query change of [APP1] default log level to Error.		Messages with log level Warn are no longer received. Messages with lower log level are still coming in.
Step 6	[LT Tester] Send request to query change of [APP1] default log level to Fatal.		Messages with log level Error are no longer received. Messages with lower log level are still coming in.





Step 7	[LT Tester] Send request to query change of [APP1] default log level to Off.	No log messages are received.
---------------	---	-------------------------------

7.2.2 [STS_LT_00002] Receiving of log messages from LT modules of several ECUs.

Test Objective	Verification that all log messages from multiple ECUs are received and that they carry the correct attributes like Application ID and ECU ID.		
ID	STS_LT_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Logging and Tracing	State	Draft
Trace to RS Criteria	RS traceability will be added in next release		
Reference to Test Environment	STC_LT_00001		
Configuration Parameters	<ul style="list-style-type: none"> - LT modules in both ECUs are configured properly. - ECU ID for [ECU1] is set to ECU1 - [APP1] has LT Application ID APPID1. - Context ID for [APP1] is set to CTX1 - ECU ID for [ECU2] is set to ECU2 - [APP2] has LT Application ID APPID2. - Context ID for [APP2] is set to CTX2 		
Summary	The LT Tester has to connect to the LT modules on the different ECUs. These have to receive and forward the log messages from the different applications in the Application Layers. First, log messages from [ECU1] on all log levels with correct attributes are expected. Then a connection to [ECU2] is established and additional messages with correct attributes are expected.		
Pre-conditions	<ul style="list-style-type: none"> - LT Tester is connected to [ECU1] via TCP socket on Port 3490. - [APP1] default log level is set to Verbose. - [APP2] default log level is set to Verbose. 		
Post-conditions	TCP connections between Jenkins server and both ECUs are closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[LT Tester] Receive log messages.	Tester receives log messages every 0.5 seconds. The messages are received for all log levels in context with ID CTX1 and contain ECU ID ECU1, and Application ID APPID1.	
Step 2	[LT Tester] Second LT Client connects to [ECU2] on Port 3490 using TCP.	Client connected.	
Step 3	[LT Tester] Receive log messages	Messages from [ECU1] are still received every 0.5 seconds. Tester additionally receives log messages from ECU2 every 0.5 seconds.	



△

		<p>△</p> <p>The additional messages are received for log level Verbose in context with ID CTX2 and contain ECU ID ECU2, and Application ID APPID2.</p>
--	--	--

8 Test configuration and test steps for Persistency

8.1 Test System

8.1.1 Test configurations

Configuration ID	STC_PER_00001
Description	Standard Jenkins server for Persistency test
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Persistency Tester is connected via Ethernet to ECU1 hosting the Persistency Test Application (PTA). The Persistency Tester is supposed to check the pass criteria.

The communication with the PTA may take place over the Diagnostics functional cluster in form of diagnostic messages. The functionality of the PTA described in the test steps may for example entirely be contained in routines that are implementation of subroutines of instances of the Diagnostic service RoutineControl. This service also provides a means to transport data from the Persistency Tester to the PTA and vice versa.

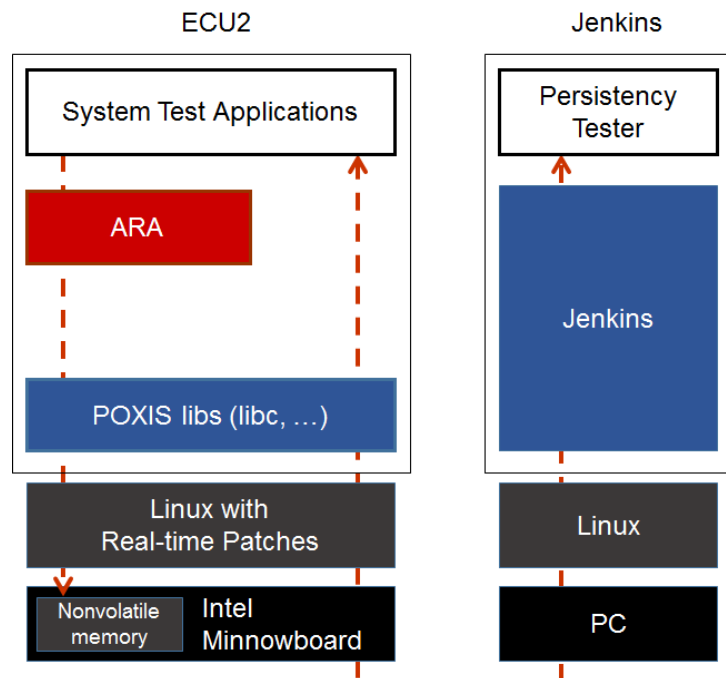


Figure 8.1: Illustration of test setup for Persistency.

8.2 Test cases

8.2.1 [STS_PER_00001] Storing an integer in a key-value database.

Test Objective	Verification, that integer data can be stored in a key-value database and that it can be retrieved again, using the associated key.		
ID	STS_PER_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00010, RS_PER_00003		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	- File system contains an empty file for the key-value database.		
Summary	Integer data is stored in a key-value database. It is then retrieved again from the database using the associated key and the retrieved value is compared to the original one.		
Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File for key-value database opened successfully and the file should be empty 		
Post-conditions	TCP connection between Persistency Tester and ECU1 is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[PTA] Store integer <intData> with associated key <intKey> in key-value database.		
Step 2	[PTA] Retrieve integer from key-value database using the associated key and store it in variable <retIntData>.		Originally written integer value is returned. And values of <intData> and <retIntData> are equal.

8.2.2 [STS_PER_00002] Storing a float in a key-value database.

Test Objective	Verification that float data can be stored in a key-value database and that it can be retrieved again, using the associated key.		
ID	STS_PER_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00003, RS_PER_00010		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	- File system contains an empty file for the key-value database.		
Summary	Float data is stored in a key-value database. It is then retrieved again from the database using the associated key and the retrieved value is compared to the original one.		
Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File for key-value database opened successfully and the file should be empty 		





Post-conditions	TCP connection between Jenkins server and ECU1 is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[PTA] Store float <floatData> with associated key <floatKey> in key-value database.	
Step 2	[PTA] Retrieve float from key-value database using the associated key and store it in variable <retFloatData>.	Originally written float value is returned. And Values of <floatData> and <ret FloatData> are equal

8.2.3 [STS_PER_00003] Storing a string in a key-value database.

Test Objective	Verification that string data can be stored in a key-value database and that it can be retrieved again, using the associated key.		
ID	STS_PER_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00003, RS_PER_00010		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	- File system contains an empty file for the key-value database.		
Summary	A string is stored in a key-value database. It is then retrieved again from the database using the associated key and the retrieved value is compared to the original one.		
Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File for key-value database opened successfully and the file should be empty 		
Post-conditions	TCP connection between Jenkins server and ECU1 is closed.		
Main Test Execution			
Test Steps			Pass Criteria
Step 1	[PTA] Store string <stringData> with associated key <stringKey> in key-value database.		
Step 2	[PTA] Retrieve string from key-value database using the associated key and store it in variable <retStringData>.	Originally written string value is returned. And Values of <stringData> and <ret StringData> are equal.	

8.2.4 [STS_PER_00004] Storing a string in a file.

Test Objective	Verification that a string can be stored in a file and retrieved again, using a file stream.		
ID	STS_PER_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00010, RS_PER_00004		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	File system contains an empty file for the file stream.		
Summary	A string is stored in a file, using a file stream. It is then retrieved again from the file and the retrieved value is compared to the original one.		
Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File stream successfully opened file and the file should be empty 		
Post-conditions	TCP connection between Jenkins server and ECU1 is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[PTA] Write string <stringData> to file via file stream.		
Step 2	[PTA] Close file.		
Step 3	[PTA] Open file.	File opened successfully.	
Step 4	[PTA] Retrieve string from file via file stream and store it in variable <retStringData>.	Originally written string value is retrieved. And Values of <stringData> and <retStringData> are equal.	

8.2.5 [STS_PER_00005] Storing an integer in a key-value database and retrieving it after reboot.

Test Objective	Verification, that integer data can be stored in a key-value database and, after a reboot, retrieved again using the associated key.		
ID	STS_PER_00005	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00001, RS_PER_00002		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	File system contains an empty file for the key-value database.		
Summary	Integer data is stored in a key-value database. A reboot is performed and the integer data is retrieved again from the database. The retrieved value is then compared to the original one.		





Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File for key-value database opened successfully and the file should be empty 	
Post-conditions	TCP connection between Jenkins server and ECU1 is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[PTA] Store integer <intData> with associated key <intKey> in key-value database.	
Step 2	[Persistency Tester] Request reboot.	
Step 3	[Persistency Tester] Wait until ECU1 has rebooted and PTA is initialized.	
Step 4	[PTA] Open database.	Database file is opened.
Step 5	[PTA] Retrieve integer from key-value database using the associated key and store it in variable <retIntData>.	Originally written integer value is returned. And Values of <intData> and <retIntData> are equal.

8.2.6 [STS_PER_00006] Storing a string in a file and retrieving it after reboot.

Test Objective	Verification, that string data can be stored in a file and, after a reboot, retrieved again using a file stream.		
ID	STS_PER_00006	AUTOSAR Releases	R18-10
Affected Functional Cluster	Persistency	State	Draft
Trace to RS Criteria	RS_PER_00001, RS_PER_00002, RS_PER_00004		
Reference to Test Environment	STC_PER_00001		
Configuration Parameters	File system contains an empty file for the file stream.		
Summary	String data is stored in a file using a file stream provided by the Persistency Functional Cluster. A reboot is performed and the string data is retrieved again from the file. The retrieved value is then compared to the original one.		
Pre-conditions	<ul style="list-style-type: none"> - Persistency tester is connected to ECU1. - Software components on ECU1 are initialized. - File stream successfully opened file and the file should be empty 		
Post-conditions	TCP connection between Jenkins server and ECU1 is closed.		
Main Test Execution			
Test Steps			Pass Criteria
Step 1	[PTA] Write string <stringData> to file via file stream.		
Step 2	[PTA] Close file.		





Step 3	[Persistency Tester] Request reboot.	
Step 4	[Persistency Tester] Wait until ECU1 has rebooted and PTA is initialized.	
Step 5	[PTA] Open file.	File opened successfully.
Step 6	[PTA] Retrieve string from file via file stream and store it in variable <retStringData>.	Originally written string value is retrieved. And Values of <stringData> and <retStringData> are equal.

9 Test configuration and test steps for Security

9.1 Test System

Identity and Access Management (IAM) requires each component to implement Policy Enforcement Point (PEP), which shall contact IAM to check access authorization of the requesting application.

System Test specification targets to check the PEP for Communication Management (FT-CM).

9.1.1 Test configurations

Configuration ID	STC_SEC_00001
Description	Standard Jenkins server for Security test
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the Security Tester is connected via Ethernet to [ECU1] hosting the Security Test Application (STA).

The Security Tester is supposed to check the pass criteria.

The communication with the STA may take place over the Diagnostics functional cluster in form of diagnostic messages.

POSIX libs (libc, ...) ARA::PER / ARA::COM Security Test Application (STA): [APP1], [APP2], [APP3] Security Tester ECU1 Linux with Real-time Patches Jenkins PC Jenkins Linux with Real-time Patches Nonvolatile Memory Intel Minnowboard

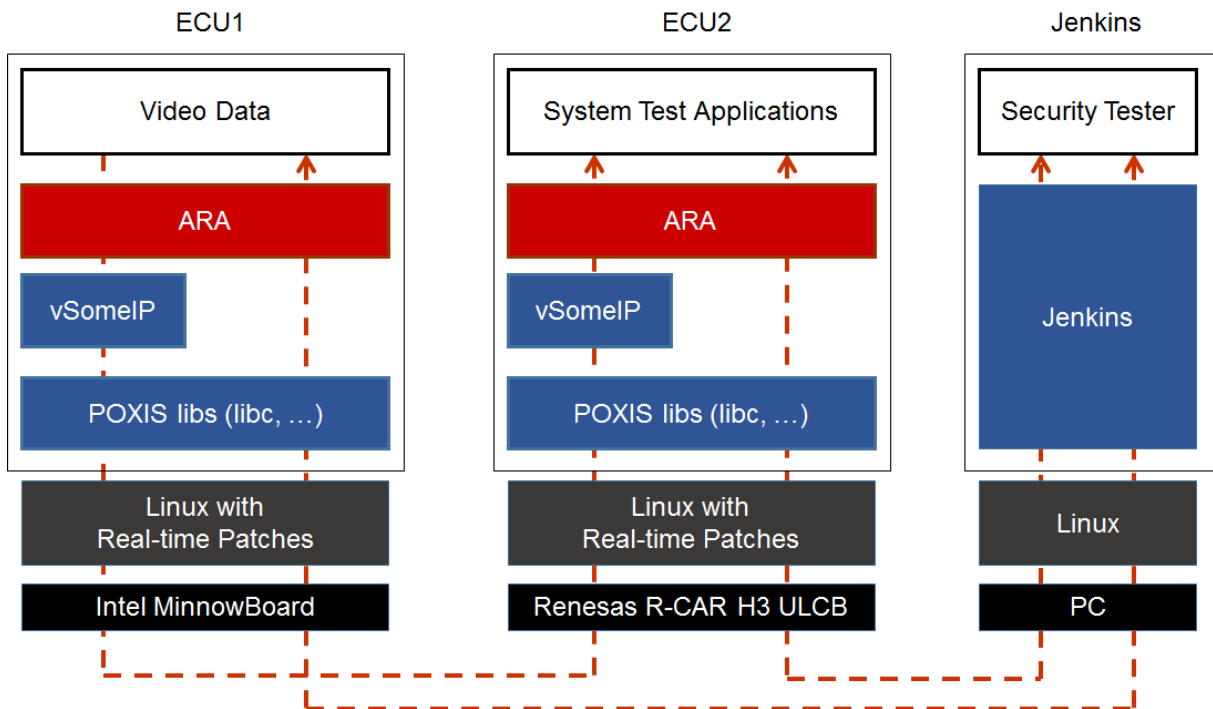


Figure 9.1: Illustration of test setup for Security.

9.2 Test cases

9.2.1 [STS_SEC_00001] Rejecting local service usage by an unauthorized application

Test Objective	Verification that unauthorized applications are not allowed to use services offered by another application.		
ID	STS_SEC_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Security	State	Draft
Trace to RS Criteria	RS_IAM_00001, RS_IAM_00002, RS_IAM_00005, RS_IAM_00006, RS_IAM_00007, RS_IAM_00010, RS_IAM_00012		
Reference to Test Environment	STC_SEC_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [APP1] offers and registers [SERVICE1], [SERVICE2], and [SERVICE3] - [APP2] is authorized to use [SERVICE2] but not [SERVICE1] and [SERVICE3] - [APP3] is authorized to use [SERVICE3] but not [SERVICE1] and [SERVICE2] 		
Summary	<ul style="list-style-type: none"> - [APP2] can successfully use [SERVICE2] but fails to use [SERVICE1] and [SERVICE3] - [APP3] can successfully use [SERVICE3] but fails to use [SERVICE1] and [SERVICE2] 		





Pre-conditions	<ul style="list-style-type: none"> - Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 	
Post-conditions	TCP connections between Security Tester and [ECU1] is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[APP1] Offers service [SERVICE1]	
Step 2	[APP1] Offers service [SERVICE2]	
Step 3	[APP1] Offers service [SERVICE3]	
Step 4	[APP2] Requests service [SERVICE2]	Service discovery callback with a handle for [SERVICE2] is received by [APP2].
Step 5	[APP3] Requests service [SERVICE3]	Service discovery callback with a handle for [SERVICE3] is received by [APP3].
Step 6	[APP2] Requests service [SERVICE1]	Service is not available.
Step 7	[APP2] Requests service [SERVICE3]	Service is not available.
Step 8	[APP3] Requests service [SERVICE1]	Service is not available.
Step 9	[APP3] Requests service [SERVICE2]	Service is not available.

9.2.2 [STS_SEC_00002] Rejecting events sent by an unauthorized application

Test Objective	Verification that unauthorized applications are not allowed to send events.		
ID	STS_SEC_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Security	State	Draft
Trace to RS Criteria	RS_IAM_00002, RS_IAM_00007, RS_IAM_00012		
Reference to Test Environment	STC_SEC_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [APP1] offers and registers [SERVICE1] and is authorized to send [EVENT11] and [EVENT12] - [APP2] offers and registers [SERVICE2] and is authorized to send [EVENT21] but not [EVENT22] - [APP3] is authorized to subscribe for [EVENT11] and [EVENT21] 		





Summary	<ul style="list-style-type: none"> - [APP1] can successfully send [EVENT11] and [EVENT12] - [APP2] can successfully send [EVENT21] but fails to send [EVENT22] - [APP3] can successfully receive [EVENT11] from [APP1] and [EVENT21] from [APP2] - [APP3] fails to receive [EVENT12] from [APP1] and [EVENT22] from [APP2] 	
Pre-conditions	<ul style="list-style-type: none"> - Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking or Driving. 	
Post-conditions	TCP connections between Security Tester and [ECU1] is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[APP1] Offers service [SERVICE1] with [EVENT11] and [EVENT12]	
Step 2	[APP2] Offers service [SERVICE2] with [EVENT21]	
Step 3	[APP3] Subscribes for [EVENT11]	Subscription is successful.
Step 4	[APP3] Subscribes for [EVENT21]	Subscription is successful.
Step 5	[APP1] Sends [EVENT11]	[APP3] receives notification for [EVENT11]
Step 6	[APP2] Sends [EVENT22]	Event is dropped silently. [APP2] is not notified.
Step 7	[APP2] Sends [EVENT21]	[APP3] receives notification for [EVENT21]
Step 8	[APP1] Sends [EVENT12]	[APP3] does not receive notification for [EVENT12]

9.2.3 [STS_SEC_00003] Rejecting events if no application is authorized to receive them

Test Objective	Verification that unauthorized applications are not allowed to receive events.		
ID	STS_SEC_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Security	State	Draft
Trace to RS Criteria	RS_IAM_00002, RS_IAM_00007, RS_IAM_00012		
Reference to Test Environment	STC_SEC_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [APP1] offers and registers [SERVICE1] and is authorized to send [EVENT11] and [EVENT12] - [APP2] offers and registers [SERVICE2] and is authorized to send [EVENT21] but not [EVENT22] - [APP3] is authorized to receive [EVENT11] 		
Summary	<ul style="list-style-type: none"> - [APP1] can successfully send [EVENT11] and [EVENT12] - [APP2] can successfully send [EVENT21] but fails to send [EVENT22] - [APP3] can successfully receive [EVENT11] from [APP1] - [APP3] fails to subscribe for [EVENT12], [EVENT21] and [EVENT22] 		





Pre-conditions	<ul style="list-style-type: none"> - Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking or Driving. 	
Post-conditions	TCP connections between Security Tester and [ECU1] is closed.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[APP1] Offers service [SERVICE1] with [EVENT11] and [EVENT12]	
Step 2	[APP2] Offers service [SERVICE2] with [EVENT21]	
Step 3	[APP3] Subscribes for [EVENT11]	Subscription is successful.
Step 4	[APP1] Sends [EVENT11]	[APP3] receives notification for [EVENT11]
Step 5	[APP1] Sends [EVENT12]	[EVENT12] is dropped and [APP3] does not receive notification for [EVENT12]
Step 6	[APP2] Sends [EVENT21]	[EVENT21] is dropped and [APP3] does not receive notification for [EVENT21]
Step 7	[APP2] Sends [EVENT22]	Event is dropped silently. [APP2] is not notified.

10 Test configuration and test steps for Update and Configuration Management

10.1 Test System

Update and Configuration Management (UCM) is responsible for Update / Installation / Uninstallation of an Adaptive Application, an Adaptive Platform itself and its underlying Operating System. There could be two use cases, Diagnostic Use case and Over The Air (OTA) Use case. System Test Specification targets to check the functionalities provided by UCM irrespective of the use cases mentioned earlier.

10.1.1 Test configurations

Configuration ID	STC_UCM_00001
Description	Standard Jenkins server for Update and Configuration Management test
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

The Jenkins Server, running the job with the UCM Tester connected via Ethernet to [ECU1] hosting the UCM Test Application (UTA).

The UCM Tester is supposed to check the pass criteria.

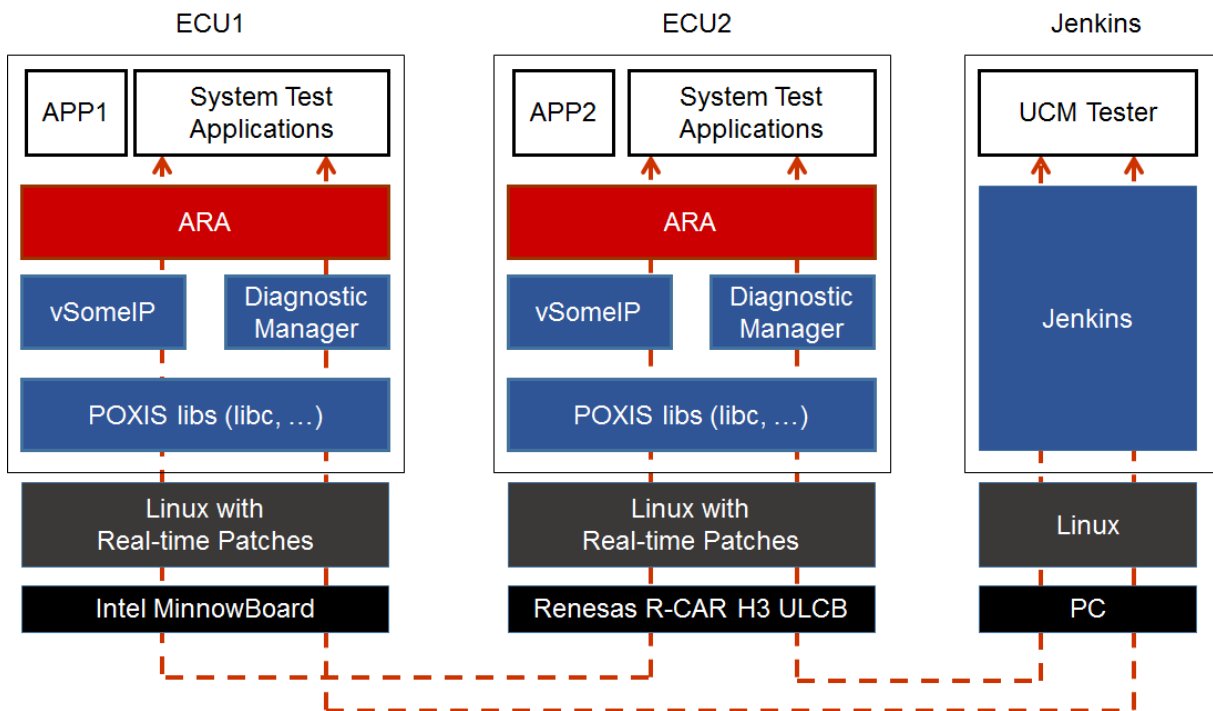


Figure 10.1: Illustration of test setup for Update and Configuration Management.

10.2 Test cases

10.2.1 [STS_UCM_00001] Check, if an update of a SW package is available.

Test Objective	Verification to check that, an Update of a SW Package is available on backend system and download the SW package, if an update is available.		
ID	STS_UCM_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Trace to RS Criteria	RS_UCM_00010 ,RS_UCM_00002		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	<ul style="list-style-type: none"> - UTA queries UCM to check Current SW version/name, UTA then queries to the backend system to check if any updated are available. If any updates are available, present the list of available SW packages to user. User then selects the required package and request UTA to download the requested package. 		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	<ul style="list-style-type: none"> - TCP connection between UCM Tester and [ECU1] is closed. 		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send a request to [UTA] to read current SW version and name from UCM		
Step 2	[UTA]: Start the mechanism to query read current SW version / name from UCM		
Step 3	[UCMTester]: Receive response from [UTA] and store it in <UCM_SWVersion>		Payload of response contains SW version and name from UCM.
Step 4	[UCMTester]: Send a request to [UTA] to read available SW version and name from Backend system		
Step 5	[UTA]: Start mechanism to read all available SW Version/Name list		
Step 6	[UCMTester]: Receive response from [UTA] and store it in <backend_SWVersion_List>		
Step 7	[UCMTester]: Send a request to download package <xyz> from available SW version/name list received from backend system.		
Step 8	[UTA]: Start mechanism to download SW package as per specified in the request.		Requested package is downloaded successfully.





Step 9	[UCMTester]: Send a request to read list of downloaded SW Packages	
Step 10	[UTA]: Start mechanism to provide list of downloaded SW packages	Downloaded SW package list is populated successfully

10.2.2 [STS_UCM_00002] Update a SW package, on user request.

Test Objective	Verification that, a SW package is updated successfully on user request		
ID	STS_UCM_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Trace to RS Criteria	RS_UCM_00011,RS_UCM_00003,RS_UCM_00023,RS_UCM_00017		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	<ul style="list-style-type: none"> - UTA has an Update available for a SW package. User selects to update the available SW package. After successful update, UTA reads SW version/name to verify that SW package is updated successfully.If update was not successful then present Failure to user. 		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. - SW Package is downloaded and available locally to be updated. 		
Post-conditions	<ul style="list-style-type: none"> - TCP connection between UCM Tester and [ECU1] is closed. 		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send request to check availability of resources for data transfer.		
Step 2	[UTA]: Start mechanism to check availability of resources.	If result == success	
Step 3	[UCMTester]: Send request(Trigger from user) to update a SW package		
Step 4	[UTA]: Starts mechanism to initialize it for approval.	Send an ACK message after successful initialization for performing an update.	
Step 5	[UCMTester]: Send request (user approval) to update a SW package as per Package manifest (SW Version and name)		
Step 6	[UTA]: Start mechanism to update a SW package.		
Step 7	[UCMTester]: Send a request to read progress status of an update.	ACK from UCM after successful update of SW package	





Step 8	[UTA]: Start mechanism to provide progress status of an update of SW package.	Current SW version/name should be equal to the SW version/name requested to be Updated
Step 9	[UCMTester]: Receive response of successful update of the package.	
Step 10	[UTA]: Read value of Persistent data associated with the SW package.	Persistent data is updated in kvs database by UCM as expected.
Step 11	[UCMTester]: Send request (user approval) to update a SW package as per Package manifest (SW version and name)	
Step 12	[UTA]: Start mechanism to update a SW package	
Step 13	[UCMTester]: Send request to read progress status of an Update.	
Step 14	[UCMTester]: Start mechanism to provide progress status of an update of the SW package	
Step 15	[UCMTester]: Receive response of unsuccessful update of the SW package.	
Step 16	[UCMTester]: Read value of Persistent data associated with the SW package.	Persistent data is not updated in KVS database by UCM

10.2.3 [STS_UCM_00003] Installing a SW package on user approval.

Test Objective	Verification that, a SW package is installed successfully on user request.		
ID	STS_UCM_00003	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Trace to RS Criteria	RS_UCM_00011,RS_UCM_00001,RS_UCM_00013,RS_UCM_00017		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	UTA has the SW package available which is to be installed. UCMTester sends user approval for installation of a SW package to UTA. UTA then queries UCM to perform SW package installation.		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			





Test Steps		Pass Criteria
Step 1	[UCMTester]: Send request to check availability of resources for data transfer	
Step 2	[UTA]: Start mechanism to check availability of resources and return Result based on availability of resource.	Result == success
Step 3	[UCMTester]: Send request (user approval) to install a SW package as per Package manifest (SW Version/name).	
Step 4	[UTA]: Start mechanism to install a SW package and write/Store Persistent data associated with the SW package.	
Step 5	[UCMTester]: Response of successful installation of package	ACK from UCM after successful installation of SW package
Step 6	[UCMTester]: Send request to read current SW version/name	SW version/name received as response should be equal to the requested SW version to be installed.
Step 7	[UTA]: Read Persistent data associated with the installed SW package from KVS database	Persistent data read is as expected .

10.2.4 [STS_UCM_00004] Uninstalling a SW package, on user request.

Test Objective	Verification that, a SW package is uninstalled successfully on user request.		
ID	STS_UCM_00004	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Trace to RS Criteria	RS_UCM_00004 ,RS_UCM_00005,RS_UCM_00018		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	UTA has the information about the SW package to be uninstalled. UCMTester sends user approval for uninstallation of a SW package to UTA. UTA then queries UCM to perform SW package uninstallation.		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send request (Trigger from user) to uninstall a SW package and Persistent data associated with the SW package as per Package manifest.		





Step 2	[UTA]: Start mechanism to uninstall a SW package.	
Step 3	[UCMTester]: Response of successful uninstallation of package	ACK from UCM after successful uninstallation of SW package
Step 4	[UCMTester]: Send request (Trigger from user) to uninstall a SW package as per package manifest	
Step 5	[UTA]: Start mechanism to uninstall a SW package	
Step 6	[UCMTester]: Response of unsuccessful installation of package	NACK from UCM after unsuccessful installation of SW package
Step 7	[UTA]: Read Persistent data associated with the uninstalled SW package	Persistent data should be deleted / not available

10.2.5 [STS_UCM_00005] Rollback to previous version, after corrupted SW package installation.

Test Objective	Verification that, a SW package is rolled back to its previous version after corrupted SW package installation on an adaptive Platform		
ID	STS_UCM_00005	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Trace to RS Criteria	RS_UCM_00008,RS_UCM_00001,RS_UCM_00023		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	- [UTA] is configured. - [Diagnostic module] is configured.		
Summary	- UCMTester queries UTA to update a SW package .Update of SW package fails.UCM informs UTA about the corruption. UTA then queries UCM to roll back to the previous working SW version.		
Pre-conditions	- UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking.		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send request to install a SW package as per Package manifest.		
Step 2	[UTA]: Start mechanism to install a SW package.		
Step 3	[UCMTester]: Send request to get SW package installation status.		





Step 4	[UTA]: Start mechanism to get Installation status of a requested SW package.	
Step 5	[UCMTester]: Receive response of installation status.	Installation status is received as Failed
Step 6	[UCMTester]: Send request to perform Rollback.	
Step 7	[UTA]: Start mechanism to rollback to latest available SW package	ACK from UCM after successful rollback.

10.2.6 [STS_UCM_00006] Read update history on an adaptive platform, on demand.

Test Objective	Verification that, an update history of an adaptive platform is available and can be read, on demand.		
ID	STS_UCM_00006	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Reference to Test Environment	STC_UCM_00001		
Trace to RS Criteria	RS_UCM_00032		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	- UTA queries UCM to read Update history, UCM checks if update history is available or not. If available, it returns update information like last update time stamp, update on user approval/auto approved.		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send request to read update history of an adaptive platform.		
Step 2	[UTA]: Start mechanism to read Update history of the platform.	ACK from UCM	
Step 3	[UCMTester]: Receive response from UTA with update history data.	Response from [UTA] regarding update history is received. Update history may contain information like-Update version , Time stamp, Previous version ,AUTO updated ,User updated etc.	
Step 4	[UCMTester]: Send request to read update history of an adaptive platform.		
Step 5	[UTA]: Start mechanism to read Update history of the platform.	NACK from UCM	





Step 6	[UCMTester]: Receive response from UTA with no history data.	Response from [UTA] regarding update history is not available.
---------------	---	--

10.2.7 [STS_UCM_00007]Data Transfer from Multiple clients,Simultaneously.

Test Objective	Verification to check that mutple clients can perform data transfer of SW Packages ,simultaneously.		
ID	STS_UCM_00007	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Reference to Test Environment	STC_UCM_00001		
Trace to RS Criteria	RS_UCM_00019		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [UTA1] is configured. - [Diagnostic module] is configured. 		
Summary	<ul style="list-style-type: none"> - UTA starts data transfer of SW package 1. - UTA1 also starts data trasfer of SW Package 2, simultaneously. - UCM allows UTA /UTA1 to perform data Trasnfer, simultaneously. 		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			
Test Steps			Pass Criteria
Step 1	[UCMTester]: Send request to UTA to transfer SW Package 1		
Step 2	[UTA]: Start mechanism to prepare for accepting SW Package 1		
Step 3	[UCMTester]: Send request to UTA1 for data transfer of SW Package 2		
Step 4	[UTA1]: Start mechanism to prepare for accepting SW Package 2		
Step 5	[UCMTester]: Send a request to get information about transferred SW Package list		
Step 6	[UTA/UTA1]: Receive response of list of SW Packages transferred to UCM		SWPackageList = SW Package 1 ,SW Package 2

10.2.8 [STS_UCM_00008]Install/Update/Removal of SW Package from multiple clients,sequentially.

Test Objective	Verification to check that mutple clients can perform Install/Update/Removal of SW packages, sequentially.		
ID	STS_UCM_00008	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Reference to Test Environment	STC_UCM_00001		
Trace to RS Criteria	RS_UCM_00024 , RS_UCM_00026,RS_UCM_00002		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [UTA1] is configured. - [Diagnostic module] is configured. 		
Summary	<ul style="list-style-type: none"> - UTA queries UCM to Install/Update/Remove SW Package 1, UTA1 also queries UCM to Install/Update/Remove SW Package 2 ,simultaneously. - UCM rejects Install/Update/Removal request from UTA1. UTA1 has to wait untill UTA finishes Install/Update/Removal of SW package 1. 		
Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		
Main Test Execution			
Test Steps		Pass Criteria	
Step 1	[UCMTester]: Send request to read current SW version.		
Step 2	[UTA]: Start mechanism to provide current SW version.		
Step 3	[UCMTester]: Receive response of current SW version and store it in <var1>.		
Step 4	[UCMTester]: Send a request to Install/Update/Remove SW Package 1 to UTA.		
Step 5	[UTA]: Start mechanism to Install/Update/Remove SW Package 1.		
Step 6	[UCMTester]: Send a request to read current SW version to UTA1		
Step 7	[UTA1]: Start mechanism to provide current SW version		
Step 8	[UCMTester]: Receive response as a SW version and store it in <var2>		
Step 9	[UCMTester]: Send a request to Install/Update/Remove SW Package 2 to UTA1		
Step 10	[UTA1]: Start mechanism to Install/Update/Remove SW package		





Step 11	[UCMTester]: Receive response as status of Install/Update/Removal	Status = Reject
Step 12	[UCMTester]: Send a request to UTA1 to get current status of UCM	
Step 13	[UTA1]: Start mechanism to provide UCM state	
Step 14	[UCMTester]: Receive response as UCM state .If State = Busy ,wait untill state changes to READY	UCMState = Busy/READY
Step 15	[UCMTester]: Send request to UTA1 to Install/Update/Removal SW Package 2	
Step 16	[UTA1]: Start mechanism to prepare for Install/Update/Removal of SW Package 2	
Step 17	[UCMTester]: Receive response as successful Install/Update/Removal of SW Package 2	
Step 18	[UCMTester]: Send a request to read SW version	
Step 19	[UTA1]: Start mechanism to send SW version of newly installed SW Package	
Step 20	[UCMTester]: Receive response as SW version of newly installed SW Package	

10.2.9 [STS_UCM_00009]Cancel Install/Update operation of SW Package .

Test Objective	Verification to check that Install/Update operation from the client can be Cancelled.		
ID	STS_UCM_00009	AUTOSAR Releases	R18-10
Affected Functional Cluster	Update and Configuration Management	State	Draft
Reference to Test Environment	STC_UCM_00001		
Trace to RS Criteria	RS_UCM_00020,RS_UCM_00002,RS_UCM_00003		
Configuration Parameters	<ul style="list-style-type: none"> - [UTA] is configured. - [Diagnostic module] is configured. 		
Summary	<ul style="list-style-type: none"> - UTA queries UCM to install/Update a SW Package 2. - UTA later realises that there are some discrepancies, it issues Cancel request to cancel ongoing Install/Update of SW Package. 		





Pre-conditions	<ul style="list-style-type: none"> - UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State Parking. 	
Post-conditions	<ul style="list-style-type: none"> - TCP connection between UCM Tester and [ECU1] is closed. 	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[UCMTester]: Send request to read current version of the installed SW Package.	
Step 2	[UTA]: Start mechanism to provide current version of SW Package.	
Step 3	[UCMTester]: Receive response of current SW version and store it in <var1>.	
Step 4	[UCMTester]: Send a request to Install/Update SW Package 2	
Step 5	[UTA]: Start mechanism to Install/Update SW Package 2	
Step 6	[UCMTester]: Send a request to cancel ongoing Install/Update of SW Package 2	
Step 7	[UTA]: Prepare to cancel ongoing operation and send an ACK for successful cancellation.	
Step 8	[UCMTester]: Send a request to read SW version.	
Step 9	[UTA]: Start mechanism to provide SW version.	
Step 10	[UCMTester]: Receive response of current SW version.	<var1> and <var2> are equal (New SW Package 2 Install/update is cancelled successfully)

11 Test configuration and test steps for E2E Protection

11.1 Test System

11.1.1 Test configurations E2E Protection

Configuration ID	STC_E2E_00001
Description	Nominal AP Apps for E2E Protection
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

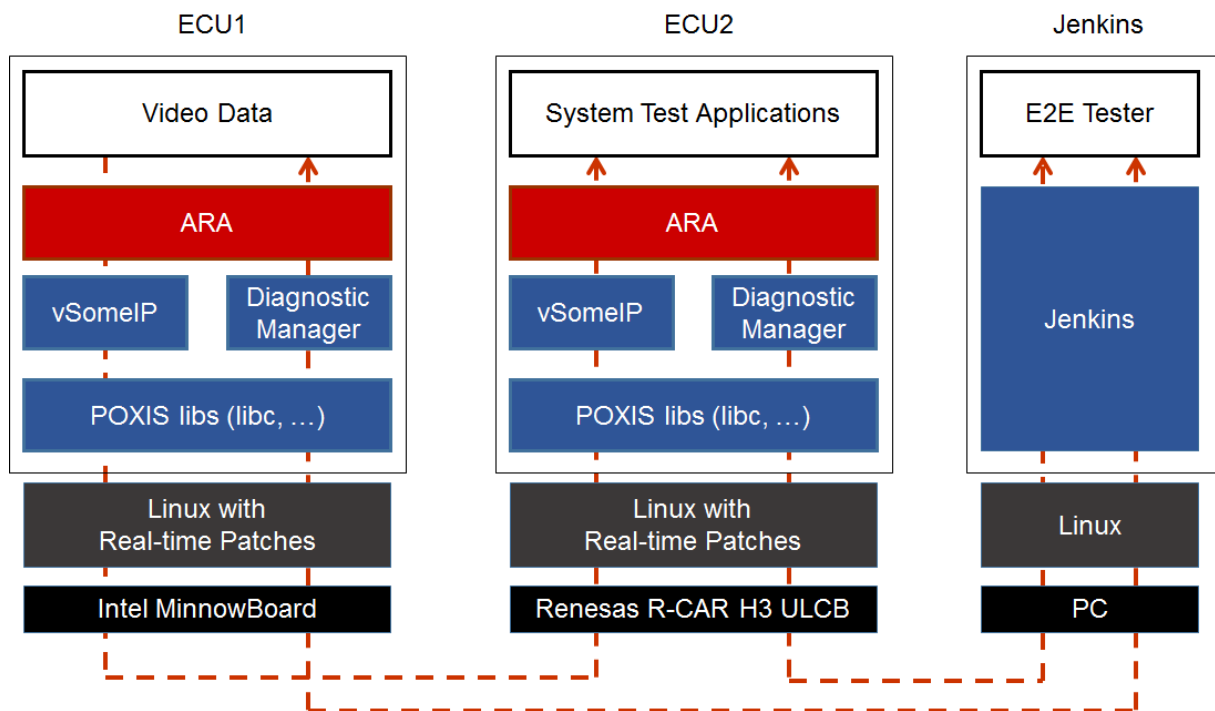


Figure 11.1: Illustration of test setup for STC-E2E-00001.

Configuration ID	STC_E2E_00002
Description	Nominal AP Apps for E2E Protection + Corrupting APP Intervention
ECU 1	Intel MinnowBoard Turbot, 192.168.100.5
ECU 2	Renesas R-Car H3 ULCB, 192.168.100.2
Jenkins	Jenkins Server, 192.168.100.10

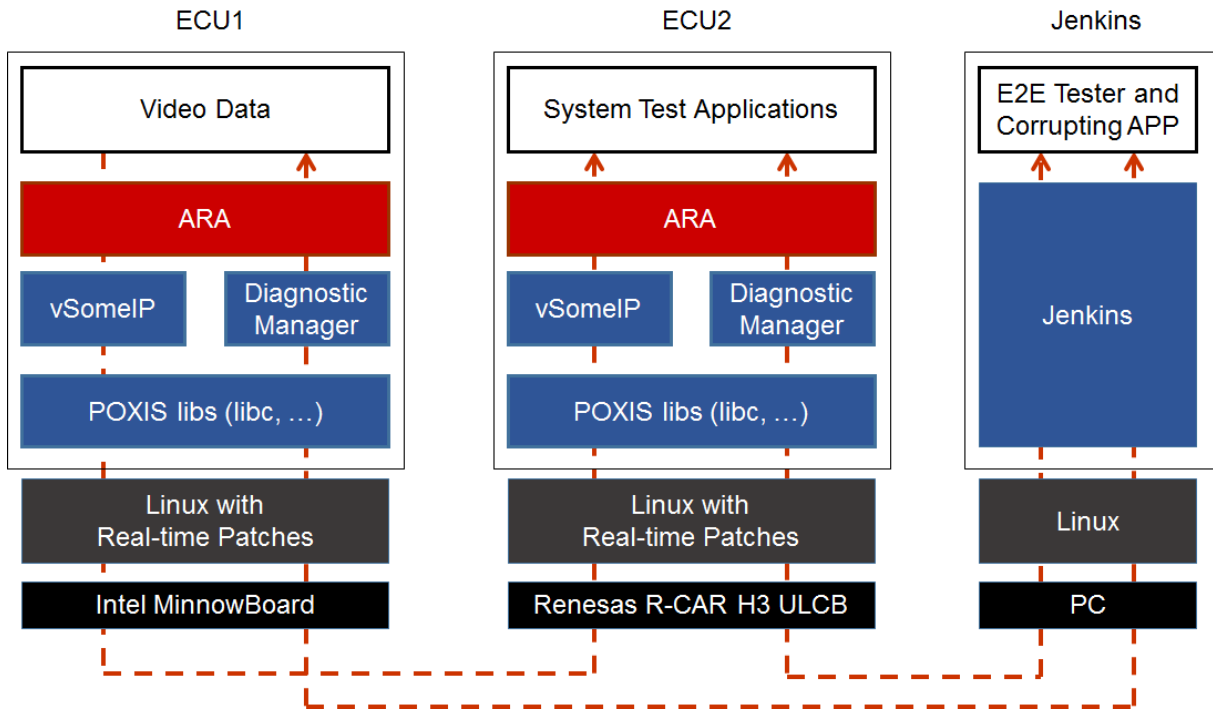


Figure 11.2: Illustration of test setup for STC-E2E-00002.

The Jenkins Server, running the job with the E2E protection test ([E2E Tester]) is connected via Ethernet to [ECU1] and [ECU2].

The [E2E Tester] is supposed to collect the results.

The communication between [E2E Tester] and the applications on ECU may take place over the Diagnostics functional cluster in form of diagnostic messages.

11.2 Test cases

11.2.1 [STS_E2E_00001] E2E Protection from AP to AP

Test Objective	To verify that the E2E protection is done properly between applications in adaptive platforms		
ID	STS_E2E_00001	AUTOSAR Releases	R18-10
Affected Functional Cluster	Safety	State	Draft
Trace to RS Criteria	RS_E2E_08541, RS_E2E_08543		
Reference to Test Environment	STC_E2E_00001		
Configuration Parameters	<ul style="list-style-type: none"> - Event based communication. - The existing communication services comprise the following (service & data names are arbitrary): - [SERVICE1]: Offered by [APP1], requested by [APP2]. - <Data1> is E2E protected, sent by [APP1], received by [APP2]. 		





Summary	[SERVICE1] & <Data1> are offered/sent by [APP1] on ECU1, and they are used/received by [APP2] on ECU2, with no problems in communication.	
Pre-conditions	<ul style="list-style-type: none"> - [E2E Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP1] and [APP2] are shut down according to Machine State. 	
Post-conditions	E2E Tester is disconnected to both ECUs.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	[SM] Request for change of Machine State to Driving from Execution Manager. Machine State for ECU1 and ECU2 are changed to Driving, and [APP 1] and [APP 2] are started up.	
Step 2	[APP1] Offer service [SERVICE1].	
Step 3	[APP2] Request service [SERVICE1].	
Step 4	[APP1] Send E2E protected <Data1> with arbitrary values	
Step 5	[APP2] Calls GetCheckStatus() for <Data1>	[APP2] reads CheckStatus = Ok
Step 6	[APP2] Executes Update for <Data1>	[APP2] receives correct value of <Data1>
Step 7	Change <Data 1> to an arbitrary value of different data length (multiple of 1 byte) and repeat steps (4->6) for 10 times	CheckStatus is always = Ok <Data1> is always received with correct values

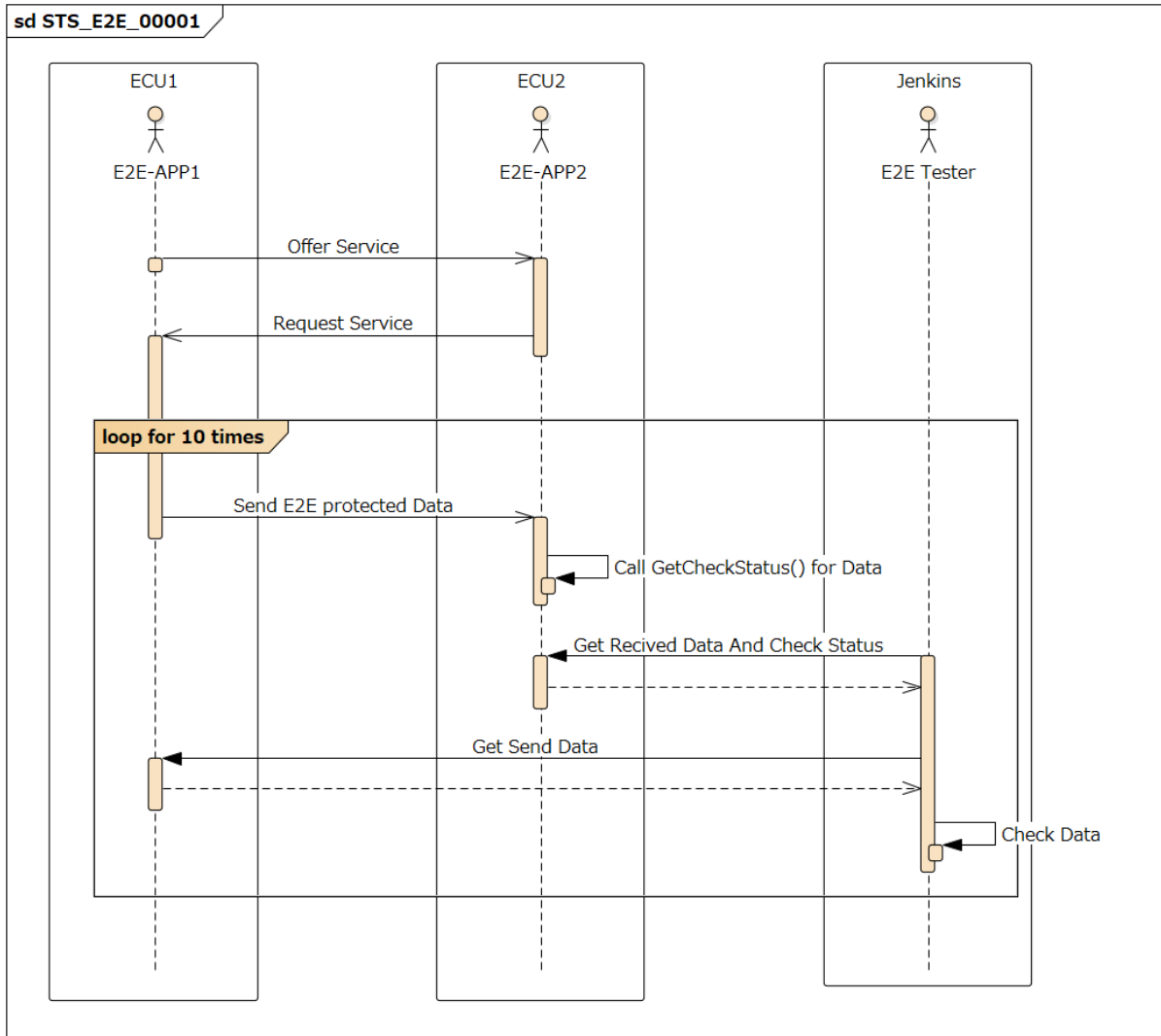


Figure 11.3: Sequence diagram of STC-E2E-00001.

11.2.2 [STS_E2E_00002] Corrupting APP Affecting Communication

Test Objective	To verify that the Corrupting APP to simulate a corrupted communication is detected by E2E		
ID	STS_E2E_00002	AUTOSAR Releases	R18-10
Affected Functional Cluster	Safety	State	Draft
Trace to RS Criteria	RS_E2E_08534		
Reference to Test Environment	STC_E2E_00002		





Configuration Parameters	<ul style="list-style-type: none"> - Event based communication. - The existing communication services comprise the following (service & data names are arbitrary): - [SERVICE1]: Offered by [APP1], requested by [APP2]. - <Data1> is E2E protected, sent by [APP1], received by [APP2]. - [Corrupting APP] to send <Data1>, with similar message format as sent by [APP1] 	
Summary	<p>[SERVICE1] & <Data1> are offered/sent by [APP1] on ECU1, and they are used/received by [APP2] on ECU2.</p> <p>When [Corrupting APP] sends the same communication sent by [APP1], but with some corrupted data, other apps detect this thanks to the E2E protection.</p>	
Pre-conditions	<ul style="list-style-type: none"> - [E2E Tester] is connected to both ECUs. - Both ECUs are in Machine State Parking. - [APP1] and [APP2] are shut down according to Machine State. 	
Post-conditions	E2E Tester is disconnected to both ECUs.	
Main Test Execution		
Test Steps		Pass Criteria
Step 1	<p>[SM]</p> <p>Request for change of Machine State to Driving from Execution Manager.</p> <p>Machine State for ECU1 and ECU2 are changed to Driving, and [APP 1] and [APP 2] are started up.</p>	
Step 2	<p>[APP1]</p> <p>Offer service [SERVICE1].</p>	
Step 3	<p>[APP2]</p> <p>Request service [SERVICE1].</p>	
Step 4	<p>[APP1]</p> <p>Send E2E protected <Data1> with arbitrary values</p>	
Step 5	<p>[APP2]</p> <p>Executes:</p> <ul style="list-style-type: none"> • GetCheckStatus() for <Data1> <p>Update for <Data1></p>	<p>[APP2]</p> <ul style="list-style-type: none"> • reads CheckStatus = Ok • receives correct value of <Data1>
Step 6	<p>[CorruptingApp]</p> <p>Sends the same Ethernet frame that was sent by [APP1], but with different value of <Data1></p>	[APP2] is notified of CRC error while receiving <Data1> (CheckStatus = Error)
Step 7	<p>[CorruptingApp]</p> <p>Sends the same Ethernet frame that was sent by [APP1], but with different data, corrupting the Counter field (The Counter should accumulate values to over DeltaCounter)</p>	[APP2] is notified of Counter error while receiving <Data1> (CheckStatus = WrongSequence)

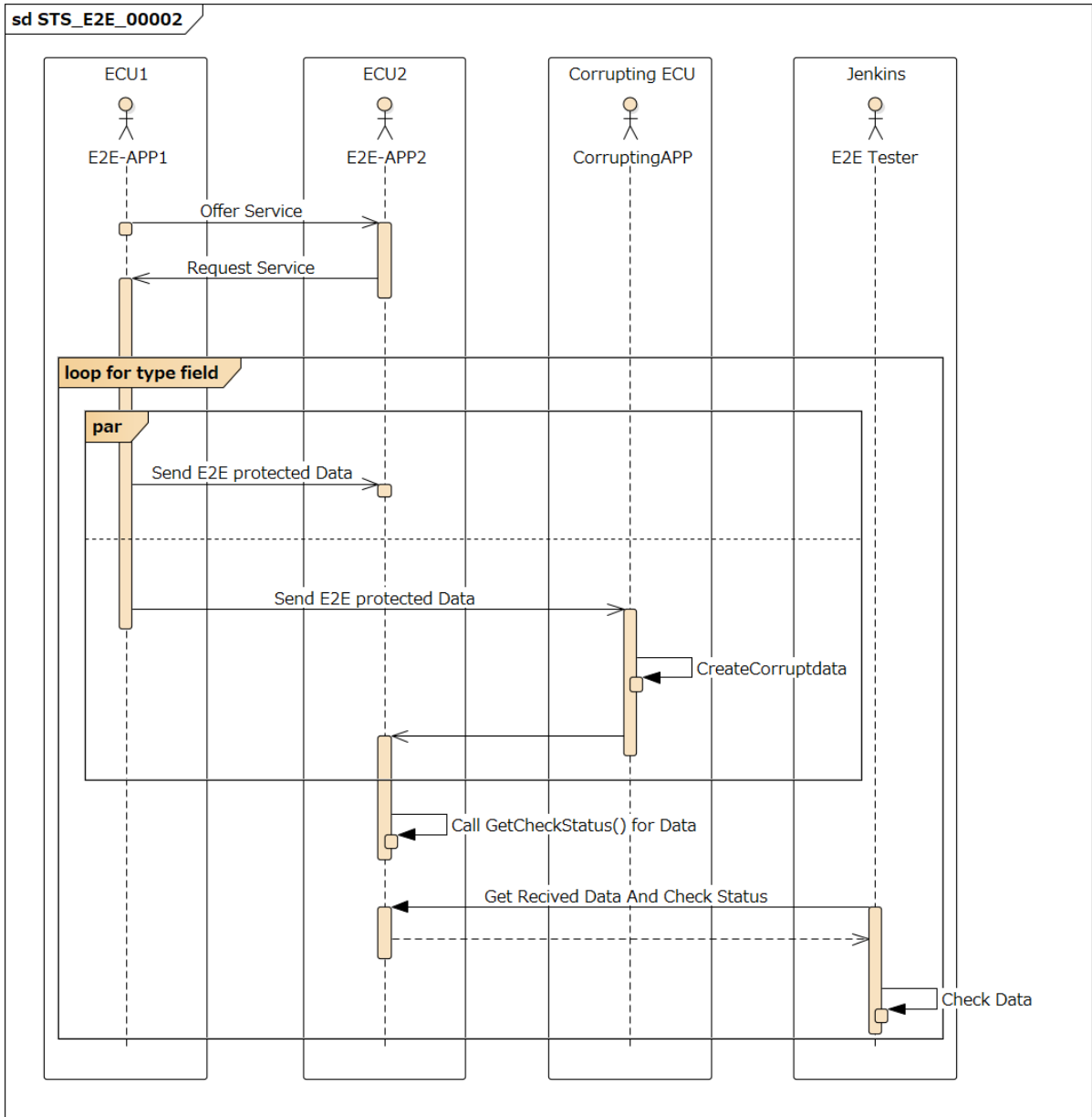


Figure 11.4: Sequence diagram of STC-E2E-00002.

12 References

[1] Glossary AUTOSAR_TR_Glossary