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	18-03

Document Change History

Date	Release	Changed by	Change Description
2018-03-29	18-03	AUTOSAR Release Management	• 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
2017-10-27	17-10	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

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

2 Scope

The following test cases are used to validate acceptance criteria in visions in order to confirm whether original intention of features are satisfied by Demonstrator of AUTOSAR Adaptive Platform.

Each test case is applicable with coupled release of specification.

3 Limitations

There are several limitations on this document.

- Test cases do not cover all of acceptance criteria which is listed on this document
- Test setup figure is not exactly same as test case description
- Test cases is not fully covered by system test implementation so far
- There are multiple way to corrupt E2E message. System test description is just one example
- Acceptance Criteria ID is not consecutive

4 Test configuration and test steps for Communication Management

4.1 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 01/09/2017.

Vision ID	AP-4	
Title	Location independence of applications	
Acceptance Criteria	**ID**	Description
	AP-4-01	Three applications in demonstrator, two locally, one remote in a one-to-n communication topology where one local application communicates with the other local and the remote application
	AP-4-02	Three applications in demonstrator, two locally, one remote in a one-to-n communication topology where one local application communicates with the other local and the remote application (see AP-4-01) swap local and remote partners from the basic setting without adaptation of the source code
	AP-4-03	Signature of the communication API does not exhibit specific items relevant for remote or local communication

Vision ID	AP-09	
Title	Discovery of local and remote Services	
Acceptance Criteria	**ID**	Description
	AP-09-01	Service offers and needs of an application can be described for the application, communicated by the application and accepted by the platform instance.
	AP-09-02	The platform instance performs service discovery and allows the involved applications to initiate communication.
	AP-09-03	Application A executed on platform instance p_1 issues provision of service S. Application B executed on platform instance p_2 issues request of service S. Platform instances interact and establish communication path between A and B via S.
	AP-09-04	Application A executed on platform instance p_1 issues provision of service S. Application B executed on platform instance p_1 issues request of service S. Platform instance establishes communication path between A and B via S.
	AP-09-05	Application A executed on platform instance p_1 issues provision of service S_1. Application B executed on platform instance p_1 issues request of service S_2. Platform instance analyses service request by B and notifies B that no service S_2 is available.
	AP-09-06	Application B executed on platform instance p_2 issues request of service S. Platform instance analyses service request by B and notifies B that no service S is available. Application A executed on platform instance p_1 issues provision of service S. Application B executed on platform instance p_2 issues second request of service S. Platform instances interact and establish communication path between A and B via S.
	AP-09-07	Static communication relationships can be defined without the need of a Service Discovery run.
	AP-09-08	Application B executed on platform instance p_2 issues request of service S. Platform instance analyses service request by B and notifies B that no service S is available. Application A executed on platform instance p_1 issues provision of service S. Platform instances interact and establish communication path between A and B via S.

	AP-09-09	Setup with two platform instances running the adaptive platform and two Adaptive Applications establish a communication by services discovery and then communicate with each other Dynamic binding: Communication relationship is established after the applications are deployed on the same ECU or on different ECUs
	AP-09-10	Setup with two platform instances running the adaptive platform and two Adaptive Applications establish a communication by services discovery and then communicate with each other Static binding: Communication relationship is established according to static definition of service binding before the applications are deployed on the same platform instance or on different platform instances.

Note: AP-09-10 Static binding part is not covered by the current system test.

Vision ID	AP-10	
Title	Service-Oriented Communication between Adaptive and Classic platform	
Acceptance Criteria	**ID**	Description
	AP-10-01	All communication will be executed via SOME/IP. Applications from the adaptive platform can communicate with other applications using service oriented communication via Ethernet.
	NOTE	ST could not ensure "all" communication is executed via SOME/IP. System test will evaluate communication can be executed via SOME/IP. Classic platform and adaptive platform communication is not the scope of System test (for R18-03).

Vision ID	AP-1601	
Title	Support of REST in Adaptive AUTOSAR	
Acceptance Criteria	**ID**	Description
	AP-1601-01	Sample application designed according to REST can be integrated and run in the demonstrator
	AP-1601-02	Serializer can process incoming data structures where not all defined elements are available
	AP-1601-03	Serializer can process incoming data structures where more than the defined elements are available
	AP-1601-04	Server responds according to a client-defined request

4.2 Test System

4.2.1 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

Configuration ID	STC_CM_00003
Description	Scenario 2 Variant 2 – Alternative Deployment
ECU 1	Renesas R-Car H3 ULCB, 192.168.100.5
ECU 2	Intel MinnowBoard Turbot, 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.

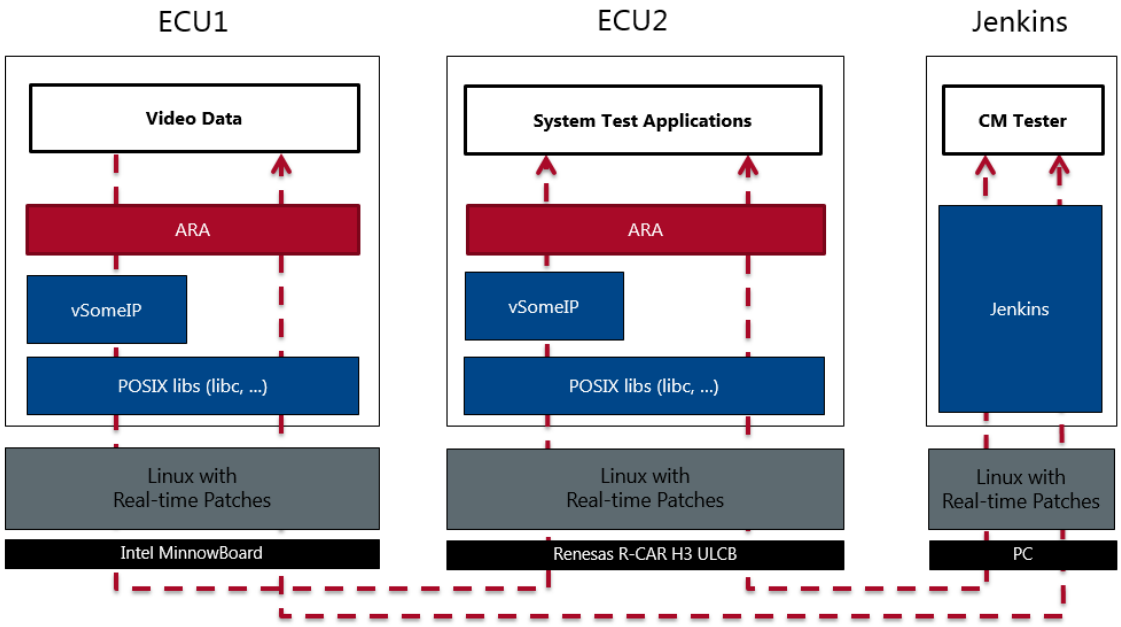

Figure 1: Illustration of test setup for Communication Management.

4.2.2 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.

4.3 Test cases Communication Management

4.3.1 [STS_CM_00001] Local and remote service discovery.

Test Objective	To verify that the applications are able to offer and request services and that service discovery works, establishing the correct communication paths.		
ID	STS_CM_00001	**AUTOSAR Releases**	R18-03
Affected Functional Cluster	Communication	**State**	Reviewed
Trace to Acceptance Criteria	AP-09-01 AP-09-02 AP-09-03 AP-09-04 AP-09-05 AP-09-06 AP-09-08 AP-10-01		
Reference to Test Environment	STC_CM_00001		
Configuration Parameters	- 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].		
Summary	First, the [APP2] and [APP3] applications on [ECU2] are started when Machine State for [ECU2] is changed to *Driving*. The [APP2] offers the services [SERVICE1] and [SERVICE2] and requests the service [SERVICE3]. [APP3] requests the service [SERVICE2] and [SERVICE4]. Then the [APP1] application on [ECU1] is started when Machine State for [ECU1] is changed to *Driving*. The [APP1] offers the service [SERVICE3] and requests the service [SERVICE1]. 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.		
Pre-conditions	- [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State *Living*. - [APP1], [APP2] and [APP3] 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	[APP3] Request service [SERVICE2].	Service is not available.
Step 7	[APP2] Offer service [SERVICE2].	
Step 8	[APP2] Request service [SERVICE3].	Service is not available.
Step 9	[CM Tester] Request change of Machine State to *Driving* for [ECU1].	Machine State for [ECU1] is changed to *Driving*.
Step 10	[APP1] Offer service [SERVICE3].	
Step 11	[APP2] Request service [SERVICE3].	Service discovery callback with a handle for service [SERVICE3] received by [APP2].
Step 12	[APP1] Request service [SERVICE1].	Service discovery callback with a handle for service [SERVICE1] is received by [APP1].
Step 13	[APP3] Request service [SERVICE4].	Service is not available.

4.3.2 [STS_CM_00002] One-to-n communication topology.

Test Objective	To verify that the applications are able to offer and request services and that service discovery as well as communication work in a one-to-n communication topology.		
ID	STS_CM_00002	**AUTOSAR Releases**	R18-03
Affected Functional Cluster	Communication Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-4-01 AP-4-03		
Reference to Test Environment	STC_CM_00002		
Configuration Parameters	- Diagnostics Read DID (brake event) and Routine Control (switch video) services configured. - The existing communication services comprise the following (service names are arbitrary): - [SERVICE5]: Offered by [APP4], requested by [APP5].		
Summary	The [APP4] application on [ECU2] offers the service [SERVICE5]. This service is		

	requested by one [APP5] instance on [ECU2] and another [APP5] instance on [ECU1]. Through successful service discovery, a one-to-n communication topology is established. [APP1] on [ECU1] is responsible for switching the video. Note: As for order of offering, no particular order of offering and requesting is necessary.
Pre-conditions	- [CM Tester] is connected to both ECUs. - Both ECUs are in Machine State *Living*. - [APP4], [APP5] on [ECU2] and [APP1], [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] 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	[CM Tester] Request [APP1] to switch to the video without brake events.	
Step 5	[CM Tester] Query [APP5] [ECU1] about the number of brake events (<BE_1A> as symbolic name).	
Step 6	[CM Tester] Query [APP5] [ECU2] about the number of brake events (<BE_2A> as symbolic name).	<BE_1A> = <BE_2A> = 0
Step 7	[CM Tester] Request [APP1] to switch to the video with brake events.	
Step 8	[APP4] Send vehicle data over [SERVICE5].	
Step 9	[CM Tester] Query [APP5 ECU1] about the number of brake events (<BE_1B> as symbolic name).	
Step 10	[CM Tester] Query [APP5 ECU2] about the number of brake events (<BE_2B> as symbolic name).	<BE_1B> = <BE_2B> and <BE_1B> = <BE_1A> +1

4.3.3 [STS_CM_00003] Swapping

The system test consists in another hardware deployment of STS_CM_00002.

Test	To verify that the deployment of applications is hardware independent on top of the

Objective	communication management (no code modification required).		
ID	STS_CM_00003	**AUTOSAR Releases**	R18-03
Affected Functional Cluster	Communication Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-4-01 AP-4-02 AP-4-03 AP-09-09		
Reference to Test Environment	STC_CM_00003		
Configuration Parameters	See [STS_CM_00002] One-to-n communication topology.		
Summary	See [STS_CM_00002] One-to-n communication topology.		
Pre-conditions	See [STS_CM_00002] One-to-n communication topology.		
Post-conditions	See [STS_CM_00002] One-to-n communication topology.		
Main Test Execution			
Test Steps		**Pass Criteria**	
Steps	See [STS_CM_00002] One-to-n communication topology.	No code modification required.	

4.4 Test cases REST

4.4.1 [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**	R18-03
Affected Functional Cluster	REST	**State**	Reviewed
Trace to Acceptance Criteria	AP-1601-01 AP-1601-04		
Reference to Test Environment	STC_REST_00001		
Configuration Parameters	RESTful API is configured		
Summary	- 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. - 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	- [REST Tester] is connected to ECU (Vehicle). - ECU is in Machine State *Living*.
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> ContentType: <application/json> Version: HTTP/1.1	
Step 2	[REST Server Application] Server Response: HTTP/1.1 200 OK 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 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	
Step 4	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status>	Positive Response is received from Server.
Step 5	[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	
Step 6	[REST Server Application]	Positive Response is

	Server Response: HTTP/1.1 200 OK Status: <*Status2*> URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status>	received from Server.
Step 7	[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 8	[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 9	[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 10	[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 11	[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 12	[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 13	[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 14	[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 15	[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 16	[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 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.

4.4.2 [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**	R18-03
Affected Functional Cluster	REST	**State**	Reviewed
Trace to Acceptance Criteria	AP-1601-01 AP-1601-04		
Reference to Test	STC_REST_00002		

Environment	
Configuration Parameters	RESTful API is configured
Summary	- 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	- [REST Tester] is connected to ECU. - ECU is in Machine State *Living* .
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 *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	
Step 2	[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <*Status5*> URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status>	Positive Response is received from Server.
Step 3	[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	
Step 4	[REST Server Application] Server Response: HTTP/1.1 200 OK URI : http://<host-name>:<port>/SERVICE2/Resource5/Element6/<Status>	Positive Response is received from Server.

Step 5	[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	
Step 6	[REST Server Application] Server Response: HTTP/1.1 200 OK Status: <*Status6*> URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status>	Positive Response is received from Server.
Step 7	[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	
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 [Element3] 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.

5 Test configuration and test steps for Execution Management

5.1 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 01/09/2017.

Vision ID	AP-60	
Title	Start and stop of Applications	
Acceptance Criteria	**ID**	Description
	AP-60-02	applications will be initiated based on specific dynamic conditions as being defined by a machine state
	AP-60-03	applications will be started at system start-up

Vision ID	AP-03	
Title	Integration of Applications	
Acceptance Criteria	**ID**	Description
	AP-03-04	When shut-down is initiated the system informs all configured and running applications and executes a well-defined shutdown sequence

5.2 Test System

5.2.1 Test configurations

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.

5.2.1.1 Machine

Machine States	*Startup (Initial Mode)*
	Shutdown
	Restart
	Driving
	Living

5.2.1.2 Application

Application Name	**APP2**		
Process	ModeDependentStartupConfig	machineMode	*Driving*
Application Name	**APP3**		
Process	ModeDependentStartupConfig	machineMode	*Driving*
Application Name	**APP4**		
Process	ModeDependentStartupConfig	machineMode	*Driving*
Application Name	**APP5**		
Process	ModeDependentStartupConfig	machineMode	*Driving*

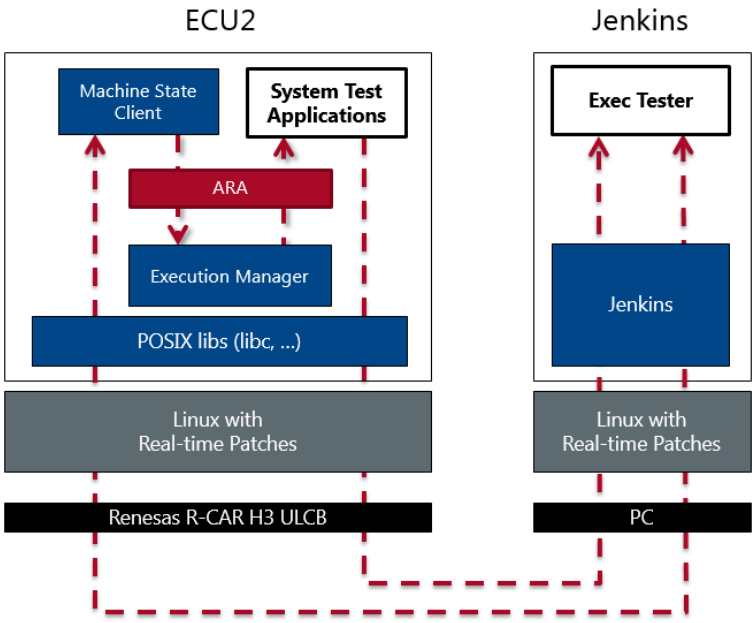

Figure 2:Illustration of test setup for Execution Management.

5.3 Test cases

5.3.1 [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**	R18-03
Affected Functional Cluster	Execution Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-60-02 AP-60-03		
Reference to Test Environment	STC_EMO_00001		

Configuration Parameters	- Machine State *Driving*, in which all System Test Applications [APP2], [APP3], [APP4] and [APP5] shall start is defined.
Summary	When initialized the system state is *Startup.* Current state is *Living.* A change of Machine State from *Living* to *Driving* is requested and the startup of the applications [APP2], [APP3], [APP4] and [APP5] associated with this Machine State is verified.
Pre-conditions	- Exec Tester is connected to ECU2 via TCP. - Software components on ECU2 are initialized. - ECU2 is in Machine State *Living*. - 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 *Driving* from Execution Manager.	Machine State for ECU2 is changed to *Driving*.
Step 3	[Exec Tester] Query execution status of [APP2].	[APP2] is executed.
Step 4	[Exec Tester] Query execution status of [APP3].	[APP3] is executed.
Step 5	[Exec Tester] Query execution status of [APP4].	[APP4] is executed.
Step 6	[Exec Tester] Query execution status of [APP5].	[APP5] is executed.

5.3.2 [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**	R18-03
Affected Functional Cluster	Execution Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-03-04		
Reference to Test Environment	STC_EMO_00002		

Configuration Parameters	- 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	When initialized the system state is *Startup.* Current state is *Driving.* A change of Machine State from *Driving* to *Shutdown* 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	- 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 *Shutdown* for ECU2.	
Step 2	[SM] Request for change of Machine State to *Shutdown* from Execution Manager.	Machine State for ECU2 is changed to *Shutdown*.
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

6 Test configuration and test steps for Diagnostics

6.1 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 04/01/2018.

Vision ID	AP-11	
Title	Support Diagnostics according to ISO 14229	
Acceptance Criteria	**ID**	Description
	AP-11-1	Diagnostics Tester is connected to vehicle network via DoIP. Diagnostics Tester establishes communication paths via vehicle port 13400 and searches for DoIP nodes on Adaptive Platform instances. Diagnostics tester selects particular DoIP node on vehicle and platform instance and requests communication connection to DoIP node for exchange of diagnostics information between Diagnostics Tester and Adaptive Platform instance.
	AP-11-2	After establishment of communication path to Adaptive Platform instance (e.g. via DoIP), Diagnostics Tester reads DTC related information from the Adaptive Platform instance.
	AP-11-3	For the application exists a Diagnostics Extract (DEXT) which defines the DTCs written by the application. The application writes DTCs, which are specified in the DEXT, in a controlled manner. All DTCs must be visible at the Diagnostics tester.
	AP-11-4	Further UDS services shall be demonstrated for acceptance.

6.2 Test System

6.2.1 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.

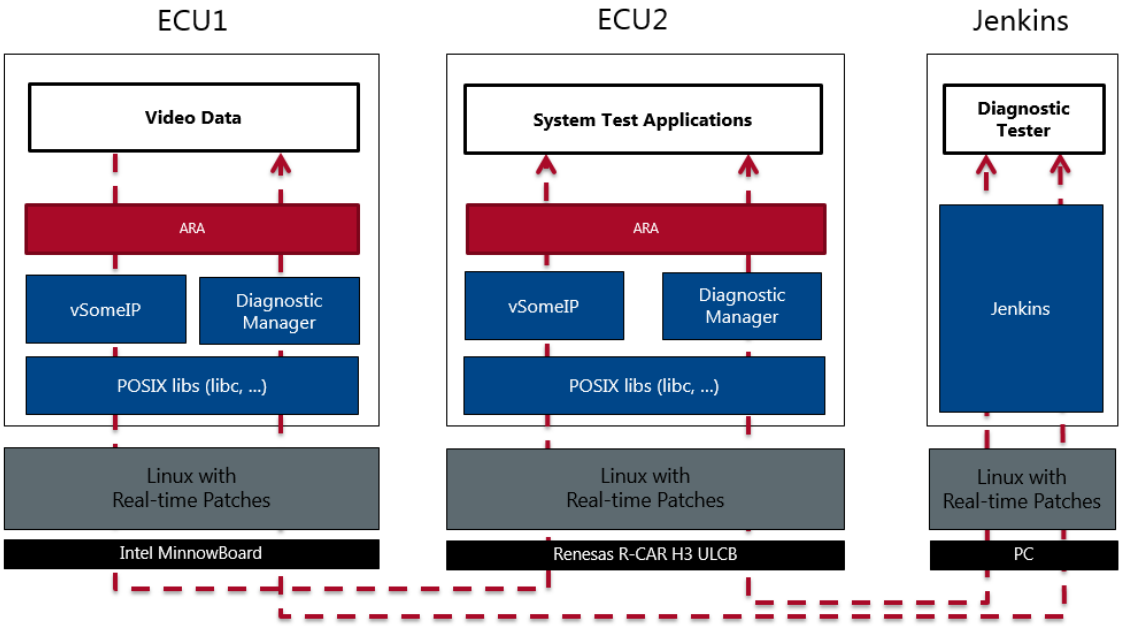

Figure 3: Illustration of test setup for Diagnostics

6.3 Test cases

6.3.1 [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**	R18-03
Affected Functional Cluster	Diagnostic	**State**	Reviewed
Trace to Acceptance Criteria	AP-11-1 AP-11-4		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	- Diagnostics module: ○ 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.

6.3.2 [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**	R18-03
Affected Functional Cluster	Diagnostic	**State**	Reviewed
Trace to Acceptance Criteria	AP-11-1 AP-11-4		
Reference	STC_DIAG_00001		

to Test Environment	
Configuration Parameters	- The following service is configured o [SERVICE1] in [APP1] - In this [SERVICE1], two different contents are available o <Content1> o <Content2> - Diagnostics module: o Service instance for service RoutineControl with RID <0x0001> is configured and only available in Extended Diagnostic Session. o 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	- [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port. - Software components on [ECU1] are initialized. - [APP1] sends <Content1> via [SERVICE1].
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 …).

6.3.3 [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**	R18-03
Affected Functional Cluster	Diagnostic	**State**	Reviewed
Trace to Acceptance Criteria	AP-11-1 AP-11-4		
Reference to Test Environment	STC_DIAG_00001		
Configuration Parameters	- Diagnostics module: ○ 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 suppressPosRspMsgIndicationBit 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 suppressPosRspMsgIndicationBit 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 …).

6.3.4 [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**	R18-03
Affected Functional	Diagnostic	**State**	Reviewed

Cluster			
Trace to Acceptance Criteria	AP-11-1 AP-11-4		
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]		Positive response received (0x62 …). Payload of

	Receive UDS response with value of <int1> and store it in <var>.	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 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 01/09/2017.

Vision ID	AP-289	
Title	Debug Log and Trace	
Acceptance Criteria	**ID**	Description
	AP-289-01	A sample application will be set up and create debug information in terms of active logs to LT. The LT viewer connected to the network shall be able to display the debug information created by the application even after the sending application has been terminated.
	AP-289-02	A sample application creates debug information and instantaneously be visible at run time by use of a LT viewer.
	AP-289-03	A sample application creates debug information at a wrong level which is in turn neither displayed at the LT viewer nor at bus level.
	AP-289-04	A LT viewer connects to the network, retrieves the available communication partners and displays the LT messages.
	AP-289-05	A LT viewer shall change the log level to retrieve and display the LT messages for the selected level.

7.2 Test System

7.2.1 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.

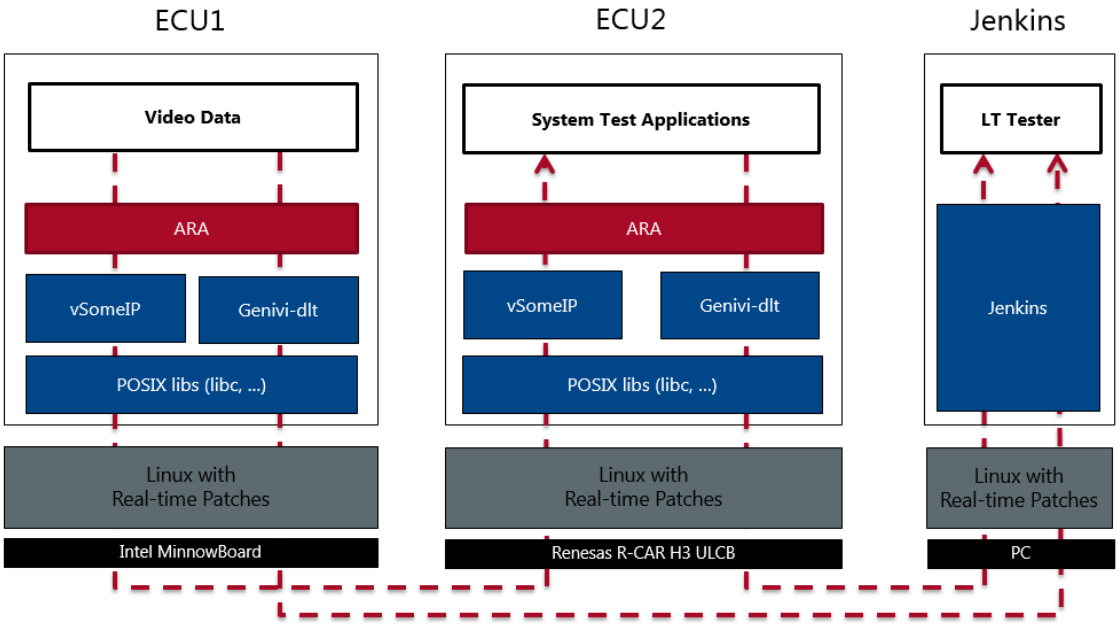

Figure 4: Illustration of test setup for Logging and Tracing.

7.3 Test cases

7.3.1 [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**	R18-03
Affected Functional Cluster	Logging and Tracing	**State**	Reviewed
Trace to Acceptance Criteria	AP-289-01 AP-289-02 AP-289-03 AP-289-04 AP-289-05		
Reference to Test Environment	STC_LT_00001		
Configuration Parameters	- 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. - 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.

7.3.2 [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**	R18-03
Affected Functional Cluster	Logging and Tracing	**State**	Reviewed
Trace to Acceptance Criteria	AP-289-01 AP-289-02 AP-289-04		
Reference to Test Environment	STC_LT_00001		
Configuration Parameters	- 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	- 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. The additional messages are received for log level Verbose in context with ID *CTX2* and contain ECU ID *ECU2*, and Application ID *APPID2*.

8 Test configuration and test steps for Persistency

8.1 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 01/09/2017.

Vision ID	AP-386	
Title	Persistent storage of data	
Acceptance Criteria	**ID**	Description
	AP-386-01	As an application developer, I want the application to store and retrieve data on a platform instance persistent over boot and ignition cycles.
	AP-386-02	As an application developer I want to access data using an unique identifier, e.g. identify a value by a key.

8.2 Test System

8.2.1 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.

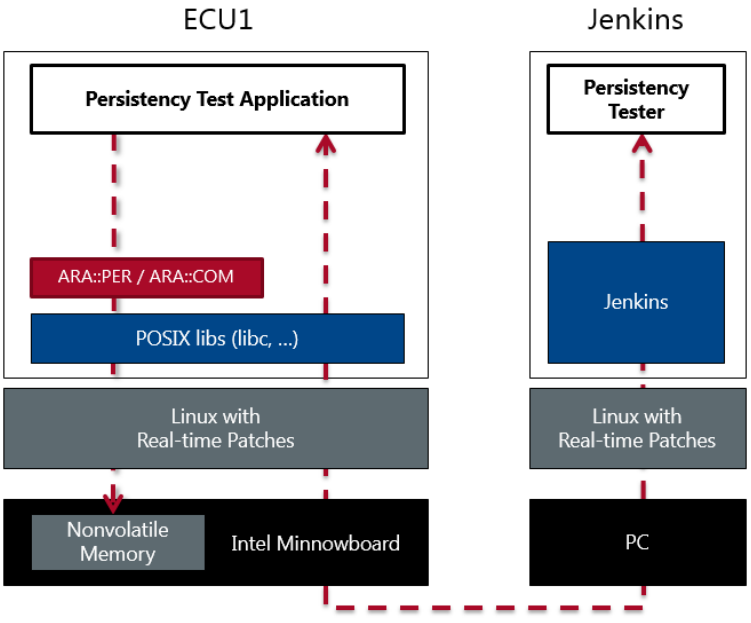

Figure 5: Illustration of test setup for Persistency.

8.3 Test cases

8.3.1 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to Acceptance Criteria	AP-386-02		
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	- 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.

8.3.2 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to Acceptance Criteria	AP-386-02		
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	- 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 <retFloatData> are equal

8.3.3 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to Acceptance Criteria	AP-386-02		
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	- 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]		Originally written string value is

	Retrieve string from key-value database using the associated key and store it in variable <retStringData>.	returned. And Values of <stringData> and <retStringData> are equal.

8.3.4 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to Acceptance Criteria	AP-386-01		
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	- 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.

8.3.5 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to	AP-386-01, AP-386-02		

Acceptance Criteria		
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	- 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.

8.3.6 [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**	R18-03
Affected Functional Cluster	Persistency	**State**	Reviewed
Trace to Acceptance Criteria	AP-386-01		
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	- 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 Acceptance criteria

System tests for SECURITY (FT-SEC) target following features:

Feature	Epic	Corresponding vision(s)
Identity and access management	AP-1137	AP-1255, AP-1257
Crypto API	AP-1623	AP-1620, AP-1621, AP-1256, AP-1257
Secure communication for SOME/IP	AP-1796	AP-1257
Secure Service Discovery using SecOC	AP-1801	AP-1257

The following acceptance criteria represent a snapshot from the JIRA system on 20/11/2017.

Vision ID	AP-1255	
Title	User Authentication & Certificates	
Acceptance Criteria	**ID**	Description
	AP-1255-04	Prove that measures are effective for listed threats

Vision ID	AP-1257	
Title	Secure Communication	
Acceptance Criteria	**ID**	Description
	AP-1257-03	Prove that measures are effective for identified threats

9.2 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).

9.2.1 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.

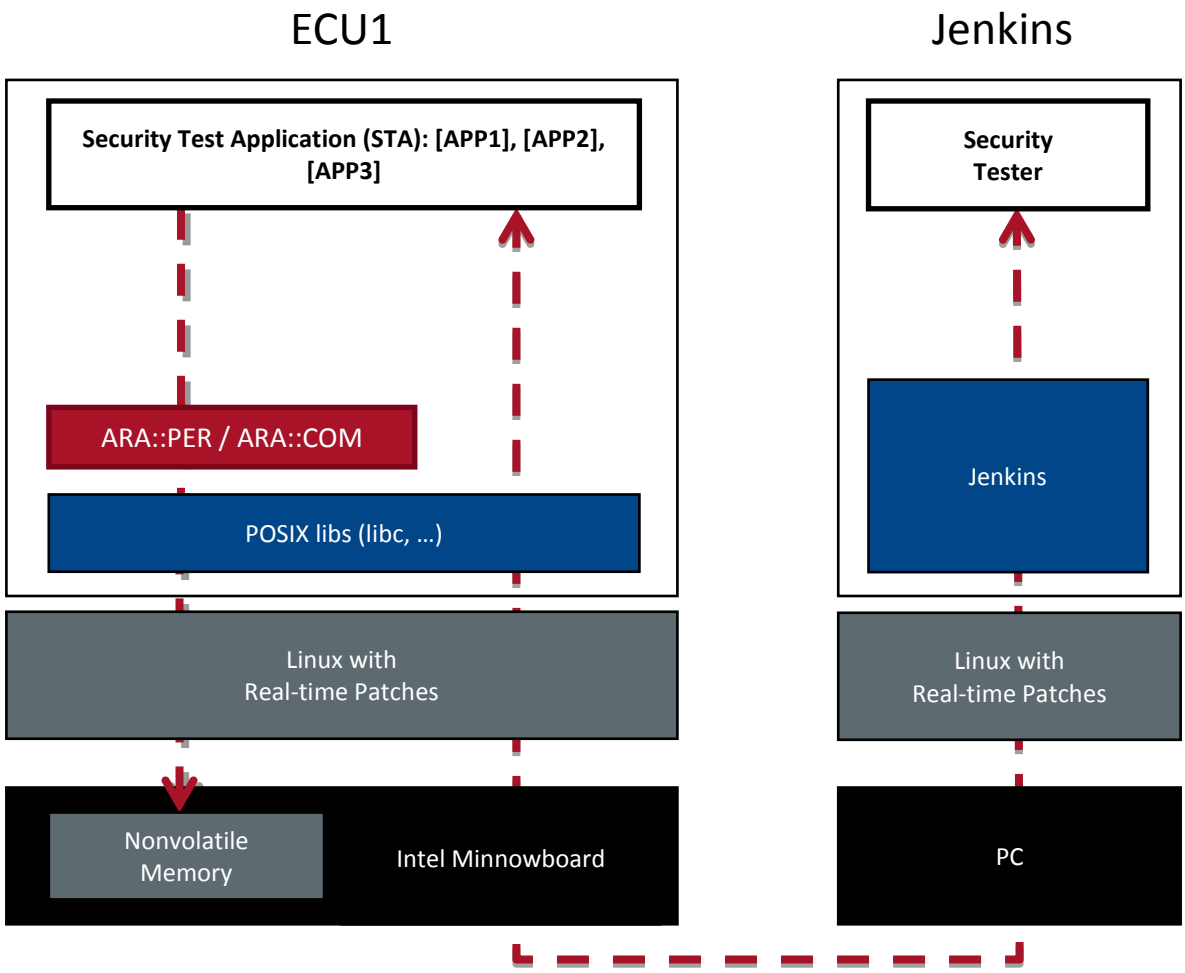

Figure 6: Illustration of test setup for Security

9.3 Test cases

9.3.1 [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**	R18-03
Affected Functional Cluster	Security	**State**	Reviewed
Trace to Acceptance Criteria	AP-1255-04 AP-1257-03		
Reference to Test Environment	STC_SEC_00001		
Configuration Parameters	- [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	- [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	- Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *Living*.
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.

9.3.2 [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**	R18-03
Affected Functional Cluster	Security	**State**	Reviewed
Trace to Acceptance Criteria	AP-1255-04 AP-1257-03		
Reference to Test Environment	STC_SEC_00001		

Configuration Parameters	- [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	- [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	- Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *Living* 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]

9.3.3 [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**	R18-03
Affected Functional Cluster	Security	**State**	Reviewed
Trace to	AP-1255-04		

Acceptance Criteria	AP-1257-03	
Reference to Test Environment	STC_SEC_00001	
Configuration Parameters	- [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	- [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	- Security Tester is connected to [ECU1] - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *Living* 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 Acceptance criteria

Vision ID	AP-417	
Title	SW installation onto vehicle	
Acceptance Criteria	**ID**	Description
	AP-417-01	How to install/update/delete adaptive applications
	AP-417-02	How to trigger an on demand update
	AP-417-03	How to configure automatic updates
	AP-417-04	How to configure the system to be able to have all those update / install capabilities (e.g. repositories, ...)
	AP-417-05	Prove that one demand can trigger the installation of an dedicated software on different adaptive platform instances

Vision ID	AP-387	
Title	Package Management	
Acceptance Criteria	**ID**	Description
	AP-387-01	Prove that the automatic update of an adaptive platform instance works
	AP-387-02	Prove the individual update/upgrade/installation of packages on an adaptive platform
	AP-387-03	Prove that application calibration data can be protected against deletion / re-initialization when (or is reinitialized) when the dedicated Application in a package was updated / upgraded
	AP-387-04	Prove that application calibration data is allocated to the application when the dedicated application was installed via a package
	AP-387-05	Prove that an corrupted installation of a package is rolled back to the latest stable installation condition
	AP-387-06	Prove that a package can be deleted consistently
	AP-387-07	Prove that installation history is available on demand

10.2 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. Update and configuration management could not initiate any operations on its own. UCM tester (Jenkin server) queries to Diagnostic module (DM) using UDS, DM then communicates with UCM Test application [UTA]. [UTA] acts as an initiator to initiate UCM operations. [UTA] queries UCM to test the services offered by UCM. UDS request sequence from UCM tester to DM needs to be considered for Implementing system tests, this documents only covers test cases to test UCM functionality (i.e. UTA-UCM).

10.2.1 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.

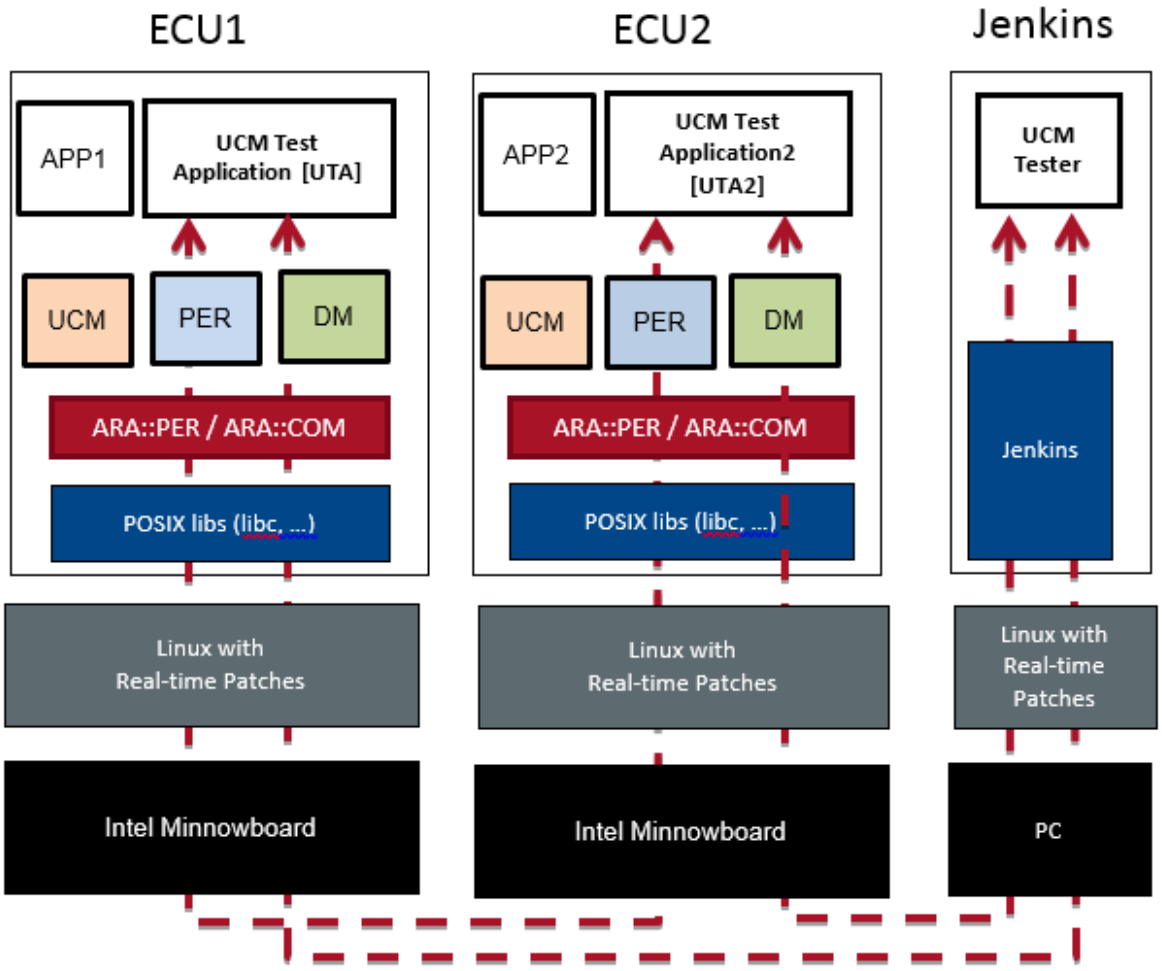

Figure 7: Illustration of test setup for Update and Configuration Management.

10.3 Test cases

10.3.1 [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**	R18-03
Affected Functional Cluster	Update and Configuration Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-417-01 AP-387-01		

Reference to Test Environment	STC_UCM_00001	
Configuration Parameters	- [UTA] is configured. - [Diagnostic module] is configured.	
Summary	- 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	- UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *"Living*".	
Post-conditions	- 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]:	Requested package is downloaded

	Start mechanism to download SW package as per specified in the request.	successfully.

10.3.2 [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**	R18-03
Affected Functional Cluster	Update and Configuration Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-417-01, AP-417-02, AP-387-02		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	- [UTA] is configured. - [Diagnostic module] is configured.		
Summary	- 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 succesfully.If update was not successful then present Failure to user.		
Pre-conditions	- UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *"Living*".		
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 update a SW package		
Step 2	[UTA]: Starts mechanism to initialize it for approval.		Send an ACK message after successful initialization for performing an update.
Step 3	[UCMTester]: Send request (user approval) to update a SW package as per Package manifest (SW Version and name)		

Step 4	[UTA]: Start mechanism to update a SW package.	
Step 5	[UCMTester]: Receive response of successful update of package	ACK from UCM after successful update of SW package
Step 6	[UCMTester]: Send request to read current SW version/name of installed SW package.	Current SW version/name should be equal to the SW version/name requested to be Updated
Step 7	[UCMTester]: Send request (user approval) to update a SW package as per Package manifest (SW Version and name)	
Step 8	[UTA]: Start mechanism to update a SW package.	NACK from UCM after unsuccessful update of SW package
Step 9	[UCMTester]: Receive response of unsuccessful update of package	

10.3.3 [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**	R18-03
Affected Functional Cluster	Update and Configuration Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-417-01, AP-417-05		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	- [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	- UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *"Living"*.		
Post-conditions	- TCP connection between UCM Tester and [ECU1] is closed.		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[UCMTester]: Send request (user approval) to install a SW package as per Package manifest (SW Version/name).	
Step 2	[UTA]: Start mechanism to install a SW package.	
Step 3	[UCMTester]: Response of successful installation of package	ACK from UCM after successful installation of SW package
Step 4	[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.

10.3.4 [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**	R18-03
Affected Functional Cluster	Update and Configuration Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-417-01, AP-387-06		
Reference to Test Environment	STC_UCM_00001		
Configuration Parameters	- [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	- UCM Tester is connected to [ECU1]. - Software components on [ECU1] are initialized. - [ECU1] is in Machine State *"Living*".		
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 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

10.3.5 [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**	R18-03
Affected Functional Cluster	Update and Configuration Management	**State**	Reviewed
Trace to Acceptance Criteria	AP-387-05		
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 *“Living*”.		
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	[UTA]: Start mechanism to rollback to latest available SW package	ACK from UCM after successful rollback.

11 Test configuration and test steps for E2E Protection

11.1 Acceptance criteria

The following acceptance criteria represent a snapshot from the JIRA system on 02/03/2018.

Vision ID	AP-1643	
Title	E2E Protection for AP	
Acceptance Criteria	**ID**	Description
	AP-1643-01	Configure safety related data for an adaptive application within the manifest with different safety levels, transmit it from an AP application and receive this data with a CP SW-C (without changing the existing SW-C on CP). Show that data was read and interpreted in the correct way.
	AP-1643-02	Configure safety related data for an adaptive application within the manifest with different safety levels, transmit it from an AP application and receive this data with an AP application. Show that data was read and interpreted in the correct way.
	AP-1643-03	Configure redundant communication paths for safety related data on different buses within the manifest and show that the receiving application or SW-C do not run into (undetected) jitter problems on redundant data.
	AP-1643-04	Configure safety and (to be) secured data that for an adaptive application within the manifest, transmit it from an AP application and receive this data with a CP SW-C (without changing the existing SW-C on CP). Show that data was read and interpreted in the correct way.
	AP-1643-05	Configure safety and (to be) secured data for an adaptive application within the manifest, transmit it from an AP application and receive this data with an AP application. Show that data was read and interpreted in the correct way.

11.2 Test System

11.2.1 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

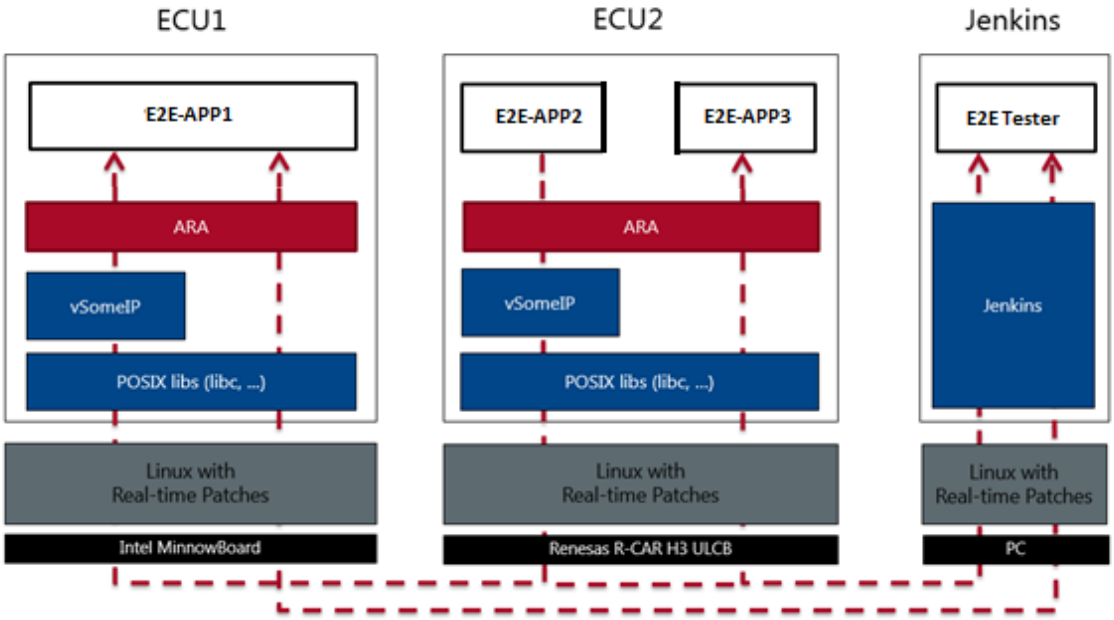

Figure 8: Illustration of test setup for STC-E2E-00001

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

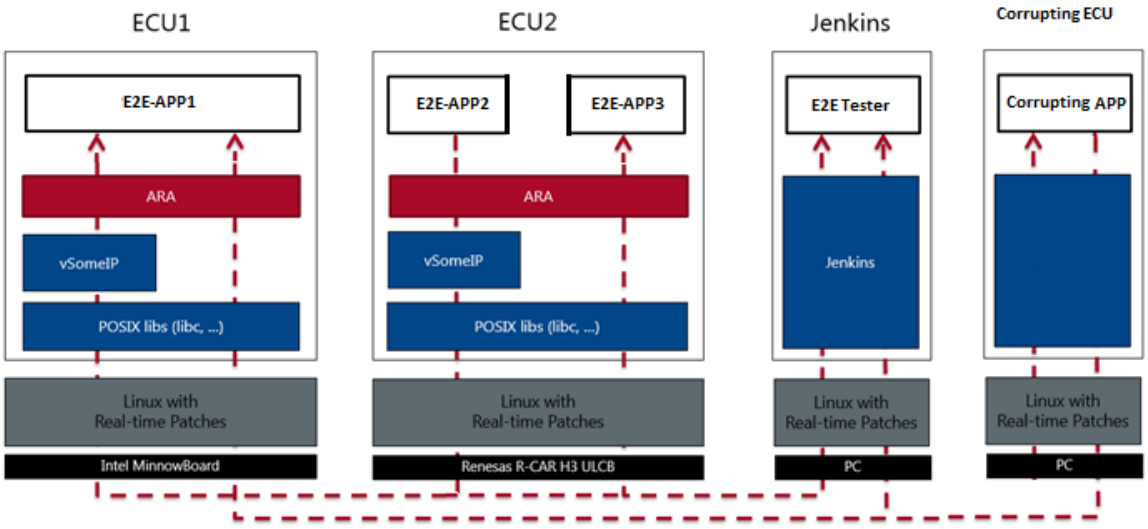

Figure 9: Illustration of test setup STC-E2E-00002

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.

11.3 Test cases E2E Protection

11.3.1 [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, in both cases (remotely and in the same ECU)		
ID	STS_E2E_00001	**AUTOSAR Releases**	R18-03
Affected Functional Cluster	Safety	**State**	Reviewed
Trace to Acceptance Criteria	AP-1643-02		
Reference to Test Environment	STC_E2E_00001		
Configuration Parameters	- The existing communication services comprise the following (service & data names are arbitrary): - [SERVICE1]: Offered by [APP2], requested by [APP1] & [APP3]. - <Data1> is E2E protected, sent by [APP2], received by [APP1] & [APP3].		
Summary	[SERVICE1] & <Data1> are offered/sent by [APP2] on ECU2, and they are used/received by [APP3] on the same ECU2, and [APP1] on another ECU1, with no problems in communication.		
Pre-conditions	- [E2E Tester] is connected to both ECUs. - Both ECUs are in Machine State *Living*. - [APP1], [APP2] and [APP3] 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	[APP2] Offer service [SERVICE1].		
Step 2	[APP3] Request service [SERVICE1].		Service discovery callback with a handle for service [SERVICE1] is received by [APP3].
Step 3	[APP1] Request service [SERVICE1].		Service discovery callback with a handle for service [SERVICE1] is received by [APP1].
Step 4	[APP2] Send E2E protected <Data1> with arbitrary values		
Step 5	[APP3] Calls GetCheckStatus() for <Data1>		[APP3] reads CheckStatus = Ok
Step 6	[APP3] Executes Update for <Data1>		[APP3] receives correct value of <Data1>
Step 7	[APP1] Calls GetCheckStatus() for <Data1>		[APP1] reads CheckStatus = Ok
Step 8	[APP1] Executes Update for <Data1>		[APP1] receives correct value of <Data1>

Step 9	Repeat steps (4->8) for 10 times with different arbitrary values of <Data1>	CheckStatus is always = Ok <Data1> is always received with correct values

11.3.2 [STS_E2E_00002] Corrupting APP Affecting Communication

Test Objective	To verify that the Corrupting APP to simulate a corrupted communication is detected by E2E (CRC or Data ID)		
ID	STS_E2E_00003	**AUTOSAR Releases**	R18-03
Affected Functional Cluster	Safety	**State**	Reviewed
Trace to Acceptance Criteria	AP-1643-02		
Reference to Test Environment	STC_E2E_00002		
Configuration Parameters	- The existing communication services comprise the following (service & data names are arbitrary): - [SERVICE1]: Offered by [APP2], requested by [APP1] & [APP3]. - <Data1> is E2E protected, sent by [APP2], received by [APP1] & [APP3]. - [Corrupting APP] to send <Data1>, with similar message format as sent by [APP2]		
Summary	[SERVICE1] & <Data1> are offered/sent by [APP2] on ECU2, and they are used/received by [APP3] on the same ECU2, and [APP1] on another ECU1. When [Corrupting APP] sends the same communication sent by [APP2], but with some corrupted data/data id, other apps detect this thanks to the E2E protection.		
Pre-conditions	- [E2E Tester] is connected to both ECUs. - Both ECUs are in Machine State *Living*. - [APP1], [APP2] and [APP3] 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	[APP2] Offer service [SERVICE1].		
Step 2	[APP3] Request service [SERVICE1].		Service discovery callback with a handle for service [SERVICE1] is received by [APP3].
Step 3	[APP1] Request service [SERVICE1].		Service discovery callback with a handle for service [SERVICE1] is received by [APP1].
Step 4	[APP2] Send E2E protected <Data1> with arbitrary values		
Step 5	[APP3] Executes: - GetCheckStatus() for <Data1> - Update for <Data1>		[APP3] - reads CheckStatus = Ok - receives correct value of <Data1>
Step 6	[APP1]		[APP1]

	Executes: - GetCheckStatus() for <Data1> Update for <Data1>	- reads CheckStatus = Ok - receives correct value of <Data1>
Step 7	[CorruptingApp] Sends the same Ethernet frame that was sent by [APP2], but with different value of <Data1>	[APP1] & [APP3] are notified of CRC error while receiving <Data1> (CheckStatus = Error) so ignoring the received value of <Data1>
Step 8	[CorruptingApp] Sends the same Ethernet frame that was sent by [APP2], but with different data, corrupting the Data ID field	[APP1] & [APP3] are notified of Data ID error while receiving <Data1> (CheckStatus = Error) so ignoring the received value of <Data1>

12 Reference

[1] Glossary

AUTOSAR_TR_Glossary