| | |
|---|---|
| **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** | 19-03 |

**Document Change History**

| Date | Release | Changed by | Description |
|---|---|---|---|
| 2019-03-29 | 19-03 | AUTOSAR Release Management | • Changed format for RS traceability items<br>• Added new section and test cases for Time Synchronization<br>• Added more test cases for CM, EMO, and DIAG |
| 2018-10-31 | 18-10 | AUTOSAR Release Management | • Added RS traceability for test cases<br>• Added ISO 9646 framework and mapping on system test architecture<br>• Added more test cases for CM, REST, EMO, and UCM |
| 2018-03-31 | 18-03 | AUTOSAR Release Management | • Test case for RESTful communication is added<br>• Test case for Security is added<br>• Test case for Update and Configuration Management is added<br>• Test case for E2E is added |
| 2010-10-27 | 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 |
| 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 R19-03 release, Requirement Specifications of CM (someip, REST), EMO, DIA, LT, PER, IAM, UCM, E2E and TS are 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.

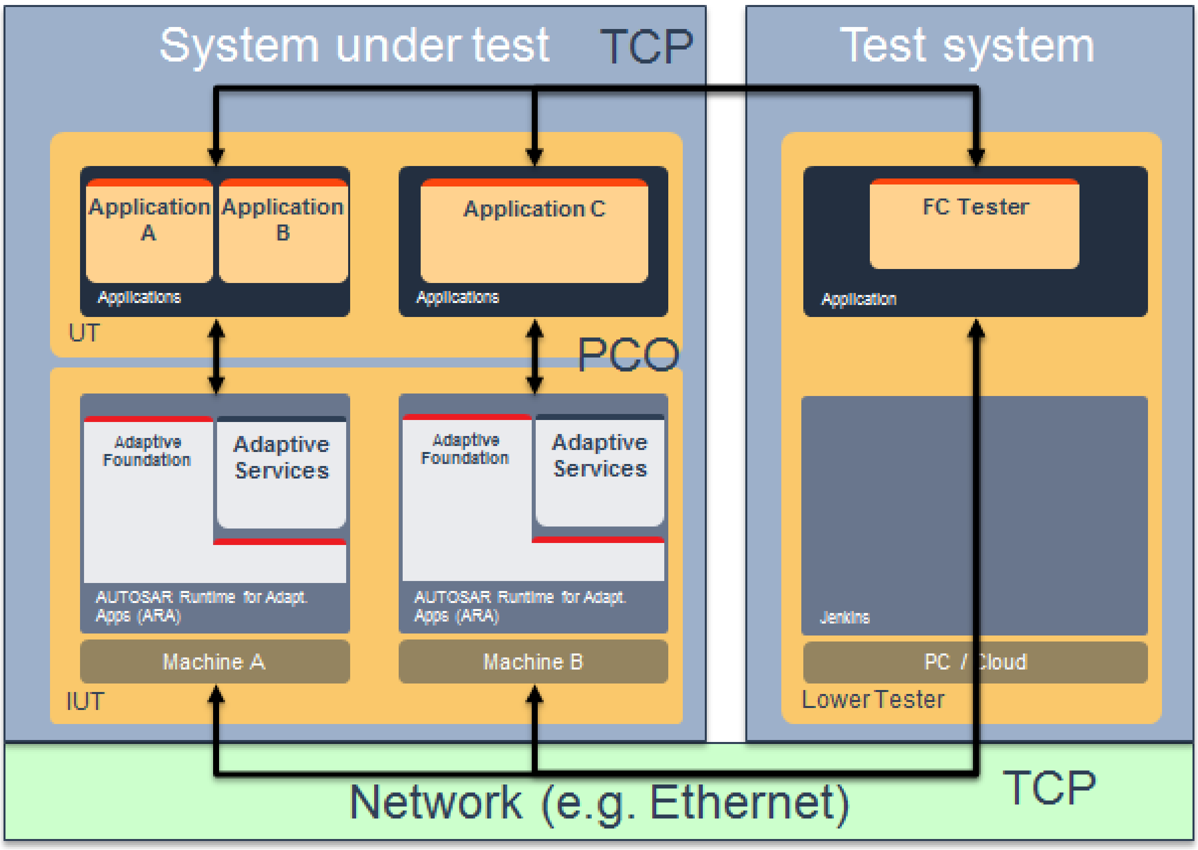

**Figure 2.1: System Test architecture**

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

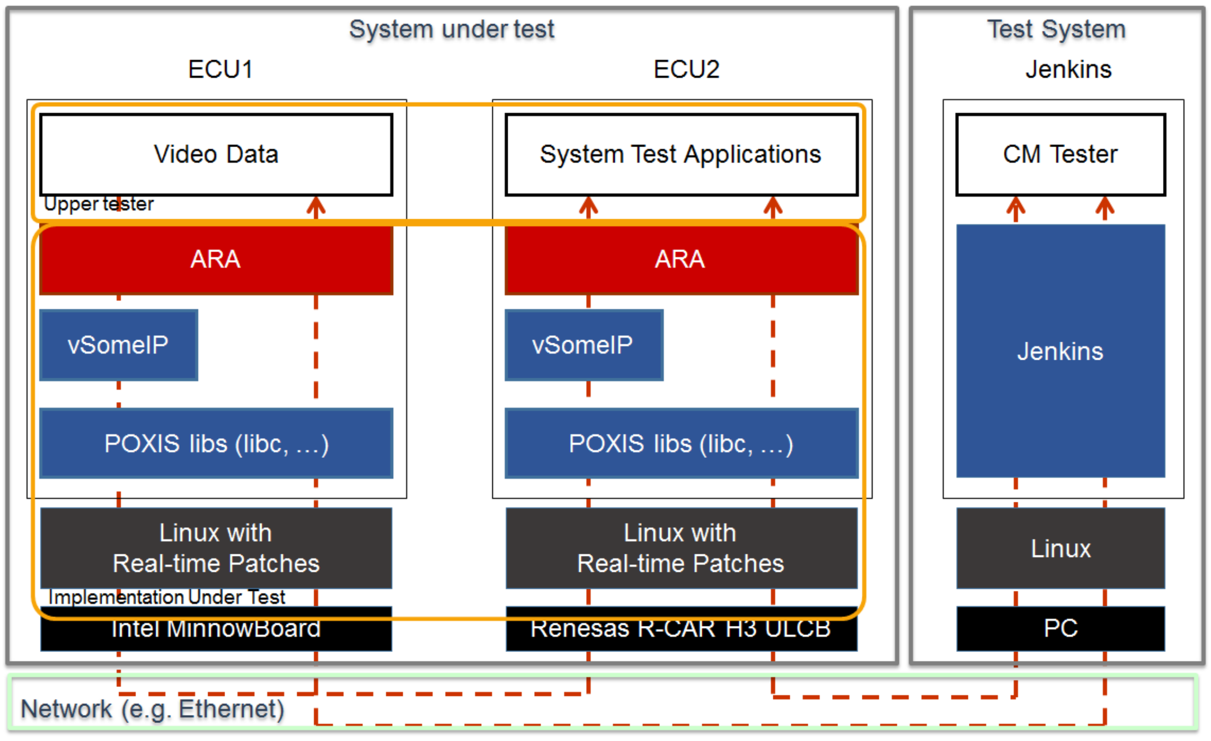

**Figure 2.2: Map to System Test implementation**

# 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
- In the E2E test case, the common parts of the E2E profiles are checked

# 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.

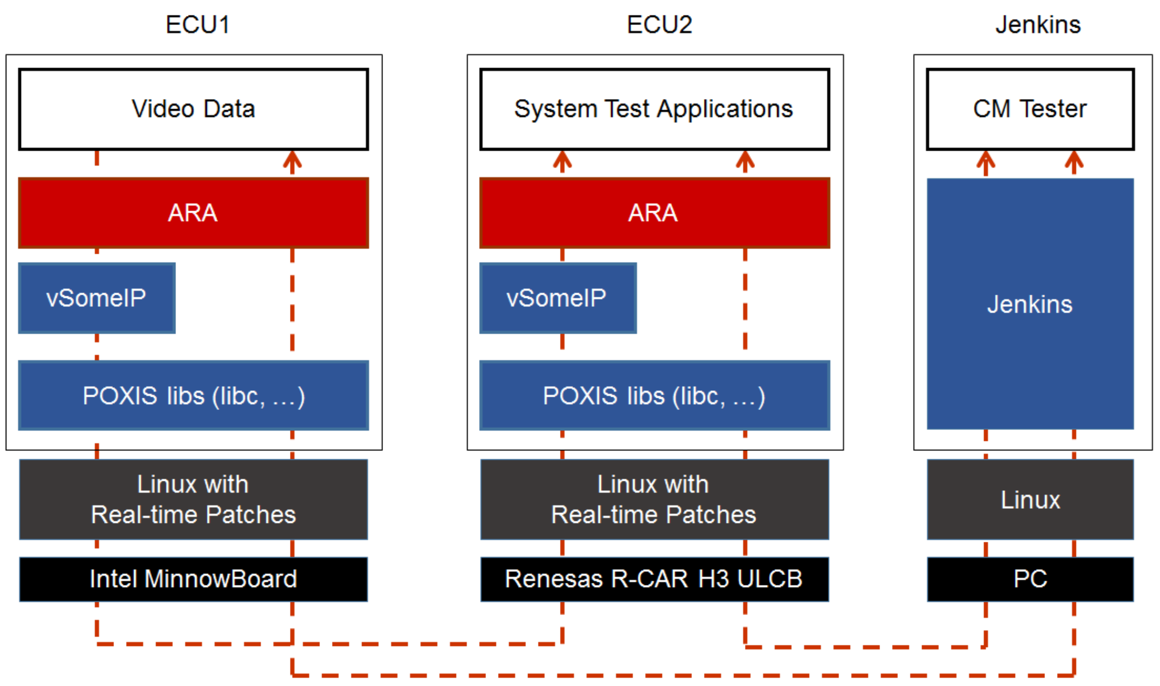

**Figure 4.1: Illustration of test setup for Communication Management**

### 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 | **State** | Draft |
| **Affected Functional Cluster** | Communication Management | | |
| **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** | - The existing communication services comprise the following (service names are arbitrary):<br>- [SERVICE1]: Offered by [APP2], requested by [APP1].<br>- [SERVICE2]: Offered by [APP2], requested by [APP3].<br>- [SERVICE3]: Offered by [APP1], requested by [APP2].<br>- [SERVICE4]: Not available, requested by [APP3].<br>- [SERVICE1], [SERVICE2], [SERVICE3] and [SERVICE4] are attributes of Methods, Events and Fields. | | |
| **Summary** | First, the [APP2] and [APP3] applications on [ECU2] are started when Machine State for [ECU2] is changed to Driving.<br>The [APP2] offers the services [SERVICE1] and [SERVICE2] and requests the service [SERVICE3].<br>[APP3] requests the service [SERVICE2].<br>The [CM Tester] trigger application [APP2] to Stop Offering service [SERVICE2].<br>Then [APP2] again offer service [SERVICE2] and initial reconnection is established between [APP2] and [APP3].<br>Then the [APP1] application on [ECU1] is started when Machine State for [ECU1] is changed to Driving.<br>The [APP1] offers the service [SERVICE3] and requests the service [SERVICE1].<br>[APP3] requests the service [SERVICE4].<br>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. | | |
| **Pre-conditions** | - [CM Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [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]<br>Request change of Machine State to Driving for [ECU2]. | Machine State for [ECU2] is changed to Driving. | |
| **Step 2** | [APP2]<br>Offer service [SERVICE1]. | | |
| **Step 3** | [APP2]<br>Offer service [SERVICE2]. | | |

▽

△

| | | |
|---|---|---|
| **Step 4** | [APP3]<br>Request service [SERVICE2]. | Service discovery callback with a handle for service [SERVICE2] is received by [APP3]. |
| **Step 5** | [CM Tester]<br>Trigger Application [APP2] to Stop Offering service [SERVICE2]. | |
| **Step 6** | [APP2]<br>Offer service [SERVICE2]. | Service discovery callback with a handle for service [SERVICE2] is received by [APP3]. |
| **Step 7** | [APP2]<br>Request service [SERVICE3]. | Service is not available. |
| **Step 8** | [CM Tester]<br>Request change of Machine State to Driving for [ECU1]. | Machine State for [ECU1] is changed to Driving. |
| **Step 9** | [APP1]<br>Offer service [SERVICE3]. | |
| **Step 10** | [APP2]<br>Request service [SERVICE3]. | Service discovery callback with a handle for service [SERVICE3] received by [APP2]. |
| **Step 11** | [APP1]<br>Request service [SERVICE1]. | Service discovery callback with a handle for service [SERVICE1] is received by [APP1]. |
| **Step 12** | [APP3]<br>Request service [SERVICE4]. | Service is not available. |
| **Step 13** | [APP1]<br>Stop offering service [SERVICE3]. | |
| **Step 14** | [APP2]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Communication Management | | |
| **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** | - The existing communication services comprise the following (service names are arbitrary):<br>- [SERVICE5]: Offered by [APP4], requested by [APP5].<br>- [SERVICE6]: Offered by [APP2], requested by [APP4].<br>- [SERVICE7]: Offered by [APP3], requested by [APP4].<br>▽ | | |

▽

△

| | | |
|---|---|---|
| | △<br>- [SERVICE5] service receives requested services synchronously.<br>- [SERVICE6] service receives requested services asynchronously. One by querying applications and another by triggering applications.<br>- [SERVICE7] service is an attribute for fire & forget methods. | |
| **Summary** | 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].<br>The [APP2] application on [ECU2] offers the service [SERVICE6]. This service is requested by one [APP4] instance on [ECU1].<br>The [APP5] on [ECU2] receives data over service [SERVICE5] from [APP4] as synchronous service call.<br>The [APP5] on [ECU1] receives data over service [SERVICE5] from [APP4] as synchronous service call.<br>The [APP4] receives data as asynchronous service call by querying application [APP2] over service [SERVICE6].<br>Then [APP4] again request service [SERVICE6].<br>The [APP3] application on [ECU2] offers service [SERVICE7]. This service is requested by one [APP4] instance on [ECU1] as fire & forget service call.<br>Then [APP4] receives data over service [SERVICE6] from [APP2] as asynchronous service call by notification.<br>Through successful service discovery, a one-to-n communication topology is established.<br>Note: As for order of offering, no particular order of offering and requesting is necessary. | |
| **Pre-conditions** | - [CM Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [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]<br>Offer service [SERVICE5]. | |
| **Step 2** | [APP5] [ECU2]<br>Request service [SERVICE5]. | Service discovery callback with a handle for service [SERVICE5] is received by [APP5] [ECU2]. |
| **Step 3** | [APP5] [ECU1]<br>Request service [SERVICE5]. | Service discovery callback with a handle for service [SERVICE5] is received by [APP5] [ECU1]. |
| **Step 4** | [APP2]<br>Offer service [SERVICE6]. | |
| **Step 5** | [APP4]<br>Request service [SERVICE6]. | Service discovery callback with a handle for service [SERVICE6] is received by [APP4] [ECU1]. |
| **Step 6** | [APP5] [ECU2]<br>Receive vehicle data over service [SERVICE5] from [APP4]. | [APP5] [ECU2]<br>Data is received from [APP4] over service [SERVICE5]. |
| **Step 7** | [APP5] [ECU1]<br>Receive vehicle data over service [SERVICE5] from [APP4]. | [APP5] [ECU1]<br>Data is received from [APP4] over service [SERVICE5]. |

▽

△

| | | |
|---|---|---|
| **Step 8** | [APP4]<br>Receive vehicle data over service [SERVICE6]. | [APP4]<br>Data is received over service [SERVICE6] by querying application [APP2] |
| **Step 9** | [APP4]<br>Request service [SERVICE6]. | Service discovery callback with a handle for service [SERVICE6] is received by [APP4] [ECU1]. |
| **Step 10** | [APP3]<br>Offer service [SERVICE7]. | |
| **Step 11** | [APP4]<br>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]<br>Receive vehicle data over service [SERVICE6]. | [APP4]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Communication Management | | |
| **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** | - The existing communication services comprise the following (service names are arbitrary):<br>- [SERVICE8]: Offered by [APP4], requested by [APP5].<br>- Service [SERVICE8] is an attribute of Events.<br>- Reception of services from Server to Proxy is possible using pooling-based style. | | |
| **Summary** | First [CM Tester] request applications on [ECU1] and [ECU2] to change Machine State to Driving.<br>[CM Tester] Request extended diagnostic session on [ECU1] and [ECU2]<br>[CM Tester] trigger application [APP4] [ECU2] to start offering service [SERVICE8] and then application [APP4][ECU2]or[ECU1] start offering service [SERVICE8].<br>Service [SERVICE8] is subscribed by application [APP5] instance on [ECU1].<br>The application [APP5] [ECU1] Queue received events, <n> being the queue length.<br>Service [SERVICE8] is subscribed by application [APP5] instance on [ECU2].<br>The application [APP5] [ECU2] Queue received events, <n> being the queue length.<br>▽ | | |

▽

△

| | | |
|---|---|---|
| | △<br>The application [APP5] [ECU1] monitors state of subscription, which is offered by [APP4] of service [SERVICE8].<br>The application [APP5] [ECU2] monitors state of subscription, which is offered by [APP4] of service [SERVICE8].<br>[CM Tester] will trigger application [APP4] [ECU1] to start sending service [SERVICE8].<br>The application [APP4] [ECU2] will send service event over service [SERVICE8].<br>The application [APP5] [ECU2] poll for receiving events from application [APP4] over service [SERVICE8].<br>The application [APP5] [ECU1] poll for receiving events from application [APP4] over service [SERVICE8].<br>[CM Tester] trigger application [APP5] [ECU2] and application [APP5] [ECU1] to stop subscribing service [SERVICE8].<br>The application [APP5] [ECU2] Monitor state of subscription from service [SERVICE8] of application [APP4].<br>The application [APP5] [ECU1] Monitor state of subscription from service [SERVICE8] of application [APP4].<br>Through successful service discovery, a one-to-n communication topology is established.<br>Note: As for order of offering, no particular order of offering and requesting is necessary. | |
| **Pre-conditions** | - [CM Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [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** | [CM Tester] Request change of Machine State to Driving for [ECU1] and [ECU2]. | |
| **Step 2** | [CM Tester]<br>Trigger Application [APP4][ECU2] to Start Offering service [SERVICE8]. | |
| **Step 3** | [APP5][ECU1]<br>Subscribe to service [SERVICE8]. | |
| **Step 4** | [APP5] [ECU1]<br>Queue received events, <n> being the queue length | |
| **Step 5** | [APP5][ECU2]<br>Subscribe to service [SERVICE8]. | |
| **Step 6** | [APP5] [ECU2]<br>Queue received events, <n> being the queue length | |
| **Step 7** | [APP5][ECU1]<br>Monitor state of subscription over service [SERVICE8]. | [APP5] [ECU1]<br>gets the current status of subscription and notification if it changes from service [SERVICE8] of application [APP4]. |
| **Step 8** | [APP5][ECU2]<br>Monitor state of subscription over service [SERVICE8]. | [APP5] [ECU2]<br>gets the current status of subscription and notification if it changes from service [SERVICE8] of application [APP4]. |

▽

△

| | | |
|---|---|---|
| **Step 9** | [CM Tester]<br><br>Trigger Application [APP4][ECU2] to Start sending service [SERVICE8]. | |
| **Step 10** | [APP4] [ECU2]<br><br>send only 10 service event [SERVICE8] | |
| **Step 11** | [APP5] [ECU2]<br><br>Poll for receiving events from application [APP4] over service [SERVICE8]. | [APP5] [ECU2]<br><br>Event is not received over service [SERVICE5] of application [APP4]. |
| **Step 12** | [APP5] [ECU1]<br><br>Poll for receiving events from application [APP4] over service [SERVICE8]. | [APP5] [ECU1]<br><br>Event is not received over service [SERVICE5] of application [APP4]. |
| **Step 13** | [CM Tester]<br><br>Trigger Application [APP5][ECU2] to Stop subscription of service [SERVICE8] | |
| **Step 14** | [CM Tester]<br><br>Trigger Application [APP5][ECU1] to Stop subscription of service [SERVICE8] | |
| **Step 15** | [APP5] [ECU2]<br><br>Monitor state of subscription from service [SERVICE8] of application [APP4]. | [APP5] [ECU2]<br><br>gets the current status of subscription, i.e. [APP5] [ECU2] has stopped subscription from service [SERVICE5]. |
| **Step 16** | [APP5] [ECU1]<br><br>Monitor state of subscription from service [SERVICE8] of application [APP4]. | [APP5] [ECU1]<br><br>gets the current status of subscription, i.e. [APP5] [ECU2] 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 | **State** | Draft |
| **Affected Functional Cluster** | Communication Management | | |
| **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** | - The existing communication services comprise the following (service names are arbitrary):<br><br>- [SERVICE5]: Offered by [APP4], requested by [APP5].<br><br>- Service [SERVICE5] is an attribute of Events.<br><br>- Reception of services from Server to Client is possible using event-based style. | | |

▽

△

| | | |
|---|---|---|
| **Summary** | First [CM Tester] request applications on [ECU1] and [ECU2] to change Machine State to Driving.<br>[CM Tester] Request extended diagnostic session [ECU1] and [ECU2].<br>[CM Tester] trigger application [APP4] [ECU1] to start offering service [SERVICE5] and then application [APP4][ECU1] start offering service [SERVICE5].<br>Service [SERVICE5] is subscribed by an application [APP5] instance on [ECU1].<br>The application [APP5] [ECU1] Queue received events, <n> being the queue length.<br>Service [SERVICE5] is subscribed by another application [APP5] instance on [ECU2].<br>The application [APP5] [ECU2] Queue received events, <n> being the queue length.<br>The application [APP5] [ECU2] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].<br>The application [APP5] [ECU1] monitors state of subscription, which is offered by [APP4] of service [SERVICE5].<br>[CM Tester] will trigger application [APP4] [ECU1] to start sending service [SERVICE5].<br>The application [APP4] [ECU1] will send service event over service [SERVICE5].<br>[APP5] [ECU2] Get triggered when receiving event over service [SERVICE5] of application [APP4]<br>[APP5] [ECU1] Get triggered when receiving event over service [SERVICE5] of application [APP4]<br>[CM Tester] trigger application [APP5] [ECU2] and application [APP5] [ECU1] to stop subscribing service [SERVICE5].<br>[APP5] [ECU1] Monitor state of subscription from service [SERVICE5] of application [APP4].<br>[APP5] [ECU2] Monitor state of subscription from service [SERVICE5] of application [APP4].<br>Through successful service discovery, a one-to-n communication topology is established.<br>Note: As for order of offering, no particular order of offering and requesting is necessary. | |
| **Pre-conditions** | - [CM Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [APP4], [APP5] on [ECU1] and [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** | [CM Tester]<br>Request change of Machine State to Driving for [ECU1] and [ECU2]. | |
| **Step 2** | [CM Tester]<br>Trigger Application [APP4][ECU1] to Start Offering service [SERVICE5]. | |
| **Step 3** | [APP5] [ECU1]<br>Subscribe to service [SERVICE5]. | |
| **Step 4** | [APP5] [ECU1]<br>Queue received events, <n> being the queue length. | |
| **Step 5** | [APP5] [ECU2]<br>Subscribe to service [SERVICE5]. | |
| **Step 6** | [APP5] [ECU2]<br>Queue received events, <n> being the queue length. | |

▽

△

| | | |
|---|---|---|
| **Step 7** | [APP5][ECU1]<br><br>Monitor state of subscription over service [SERVICE5]. | [APP5] [ECU1]<br><br>gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4]. |
| **Step 8** | [APP5][ECU2]<br><br>Monitor state of subscription over service [SERVICE5]. | [APP5] [ECU2]<br><br>gets the current status of subscription and notification if it changes from service [SERVICE5] of application [APP4]. |
| **Step 9** | [CM Tester]<br><br>Trigger Application [APP4][ECU2] to Start sending service [SERVICE5]. | |
| **Step 10** | [APP4] [ECU1]<br><br>send service event [SERVICE5]. | |
| **Step 11** | [APP5] [ECU2]<br><br>Get triggered when receiving event over service [SERVICE5] of application [APP4]. | [APP5] [ECU2]<br><br>Events received and read them at the same time from service [SERVICE5]. |
| **Step 12** | [APP5] [ECU1]<br><br>Get triggered when receiving event over service [SERVICE5]. | [APP5] [ECU1]<br><br>Events received and read them at the same time from service [SERVICE5] of application [APP4]. |
| **Step 13** | [CM Tester]<br><br>Trigger Application [APP5][ECU2] to Stop subscription of service [SERVICE5] | |
| **Step 14** | [CM Tester]<br><br>Trigger Application [APP5][ECU1] to Stop subscription of service [SERVICE5] | |
| **Step 15** | [APP5] [ECU1]<br><br>Monitor state of subscription from service [SERVICE5] of application [APP4]. | [APP5] [ECU1]<br><br>gets the current status of subscription, i.e.[APP5] [ECU1] has stopped the subscription from service [SERVICE5]. |
| **Step 16** | [APP5] [ECU2]<br><br>Monitor state of subscription from service [SERVICE5] of application [APP4]. | [APP5] [ECU2]<br><br>gets the current status of subscription, i.e.[APP5] [ECU2] has stopped the subscription from service [SERVICE5]. |

### 4.2.5 [STS_CM_00005] Communication for Fields.

| | | | |
|---|---|---|---|
| **Test Objective** | To verify that the applications are able to receive notifications as well as query (get) and modify (set) field value and that communication work for Fields. | | |
| **ID** | STS_CM_00005 | **State** | Draft |
| **Affected Functional Cluster** | Communication Management | | |

▽

△

| | | |
|---|---|---|
| **Trace to RS Criteria** | [RS_CM_00216], [RS_CM_00217], [RS_CM_00218], [RS_CM_00219], [RS_CM_00220], [RS_CM_00221] | |
| **Reference to Test Environment** | STC_CM_00001 | |
| **Configuration Parameters** | - The existing communication services comprise the following (service names are arbitrary):<br>- [SERVICE5]: Offered by [APP4], requested by [APP5]. | |
| **Summary** | Initially [CM Tester] requests applications on [ECU1] and [ECU2] to change Machine State to Driving.<br>[CM Tester] requests [APP5] to get the current field value of service [SERVICE5] [APP4].<br>In turn [APP5] requests [APP4] to get the current field value of service [SERVICE5] [APP4].<br>The [APP4] provides a method to get the current field value of service [SERVICE5] [APP4].<br>[CM Tester] requests [APP5] to set the current field value of service [SERVICE5] [APP4].<br>In turn [APP5] requests [APP4] to set the current field value of service [SERVICE5] [APP4].<br>The [APP4] provides a method to set the current field value of service [SERVICE5] [APP4].<br>[APP4] sends normal return code notification to [APP5].<br>[APP5] returns a normal return code to [CM Tester].<br>Through successful service discovery, a communication is established.<br>A field without a setter and without a getter and without a notifier shall not exist. The field shall contain at least a getter, a setter, or a notifier.<br>Note: As for order of offering, no particular order of offering and requesting is necessary. | |
| **Pre-conditions** | - [CM Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [APP4] 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** | [CM Tester]<br>Request change of Machine State to Driving for [ECU1] and [ECU2]. | |
| **Step 2** | [CM Tester]<br>Request [APP5] to get the current field value of service [SERVICE5] [APP4]. | |
| **Step 3** | [APP5]<br>Request [APP4] to get the current field value of service [SERVICE5] [APP4]. | [APP4]<br>Receives the request from application [APP5]. |
| **Step 4** | [APP4]<br>Provides a method to get the current field value of service [SERVICE5] [APP4]. | [APP5]<br>Receives response message from [APP4]. |
| **Step 5** | [APP5]<br>Returns the current field value of service [SERVICE5][APP4] to [CM Tester]. | [CM Tester]<br>Receives the default field value (e.g. zero) of [SERVICE5][APP4]. |
| **Step 6** | [CM Tester]<br>Request [APP5] to set the current field value of service [SERVICE5][APP4]. | |
| **Step 7** | [APP5]<br>Request [APP4] to set the field value of service [SERVICE5][APP4]. | [APP4]<br>Receives the request from application [APP5]. |

▽

△

| | | |
|---|---|---|
| **Step 8** | [APP4]<br><br>Provides a method to set the current field value of service [SERVICE5][APP4]. | [APP5]<br><br>Receives response message from [APP4]. |
| **Step 9** | [APP4]<br><br>sends normal return code notification to [APP5]. | [APP5]<br><br>Receives termination notification from[APP4]. |
| **Step 10** | [APP5]<br><br>returns a normal return code to CM tester | [CM Tester]<br><br>Receives termination notification from[APP4]. |
| **Step 11** | [CM Tester]<br><br>Request [APP5] to get the set field value of service [SERVICE5][APP4]. | |
| **Step 12** | [APP5]<br><br>Request [APP4] to get the current field value of service [SERVICE5] [APP4]. | [APP4]<br><br>Receives the request from application [APP5]. |
| **Step 13** | [APP4]<br><br>Provides a method to get the current field value of service [SERVICE5] [APP4]. | [APP5]<br><br>Receives response message from [APP4]. |
| **Step 14** | [APP5]<br><br>Returns the set field value of service [SERVICE5][APP4] to [CM Tester]. | [CM Tester]<br><br>Receives the set field value (set in the previous steps) of [SERVICE5][APP4]. |

## 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 | **State** | Draft |
| **Affected Functional Cluster** | REST | | |
| **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** | • Client is in backend/ cloud and server is in vehicle.<br>• First client is set up and request is created with URI and Methods<br>(GET/PUT/ POST/DELETE/OPTIONS).<br>• Request is sent and response is received from server.<br>• 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<br>▽ | | |

▽

△

| | | |
|---|---|---|
| | △<br>Element2 have <State3> and <State4>. A new element [Element3] is created in resource [Resource2] using POST and later [Element3] is deleted using DELETE.<br>• Response from server is processed and then client unsubscribe from the event.<br>• Client is stopped. | |
| **Pre-conditions** | - [REST Tester] is connected to ECU (vehicle).<br>- 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]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength : <length><br>Accept: <application/json><br>Version: HTTP/1.1 | |
| **Step 2** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Content-Type: <application/json><br>Status: <Status1><br>URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status> | Positive response is received from Server. |
| **Step 3** | [REST Client Application]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength : <length><br>Accept: <application/xml><br>Version: HTTP/1.1 | |
| **Step 4** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Content-Type: <application/xml><br>Status: <Status1><br>URI : http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status> | Positive response is received from Server. |

▽

△

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

▽

△

| | | |
|---|---|---|
| **Step 11** | [REST Client Application]<br>Send Request to get details of Resource2<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource2<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 12** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status> | Positive response is received from Server. |
| **Step 13** | [REST Client Application]<br>Send Request to create Resorce2/Element3<br>Method: POST<br>URI: http://<host-name>:<port>/SERVICE1/Resource2/Element3<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 14** | [REST Server Application]<br>Server Response: HTTP/1.1 201 Created<br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3 | Positive response is received from Server. |
| **Step 15** | [REST Client Application]<br>Send Request to get details of Resource2<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource2<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 16** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status><br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3/<Status> | Positive response is received from Server. |
| **Step 17** | [REST Client Application]<br>Send Request to delete [Element3]<br>Method: DELETE<br>URI: http://<host-name>:<port>/SERVICE1/Resource2/Element3<br>Host: <host-name><br>▽ | |

▽

△

| | | |
|---|---|---|
| | △<br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 18** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element3 | Positive response is received from Server. |
| **Step 19** | [REST Client Application]<br>Send Request to get details of Resource2<br>(Element 3 should be deleted)<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource2<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 20** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE1/Resource2/Element2/<Status> | Positive response is received from Server. |
| **Step 21** | [REST Client Application]<br>Send an invalid URI Request<br>Method = GET,<br>URI: http://<host-name>:<port>/SERVICE5 | |
| **Step 22** | [REST Server Application]<br>Server replies with Status: 404<br>URI: http://<host-name>:<port>/SERVICE5 | Negative response is received from Server. |
| **Step 23** | [REST Client Application]<br>Send multiple requests from client.<br>Method = GET,<br>URI: http://<host-name>:<port>/SERVICE1/Resource1<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1<br>Method = GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource2<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |

▽

△

| | | |
|---|---|---|
| **Step 24** | [REST Server Application]<br>Server replies with Status: 200 OK<br>URI: http://<hostname>:<port>/SERVICE1/Resource1/<status><br>Server replies with Status: 200 OK<br>URI: http://<hostname>:<port>/SERVICE1/Resource2/<status> | Positive 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 | **State** | Draft |
| **Affected Functional Cluster** | REST | | |
| **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** | - Client is in vehicle and server is in backend/ cloud.<br>- First client is set up and request is created with URI and Methods<br>(GET/PUT/ POST/DELETE/OPTIONS).<br>- Request is sent and response is received from server.<br>- 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.<br>- Response from server is processed and then client unsubscribe from the event.<br>Client is stopped. | | |
| **Pre-conditions** | - [REST Tester] is connected to ECU.<br>- 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]<br>Send Request to get status of Resource5/Element5<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/?Status<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | | |

▽

△

| | | |
|---|---|---|
| **Step 2** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Status: <Status5><br>URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status> | Positive response is received from Server. |
| **Step 3** | [REST Client Application]<br>Send Request to update Resource5/Element5<br>(change status 5 to status 6)<br>Method: PUT<br>URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/Status6<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 4** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status> | Positive response is received from Server. |
| **Step 5** | [REST Client Application]<br>Send Request to get status of Resource5/Element5<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE2/Resource5/Element5/?Status<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 6** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Status: <Status6><br>URI : http://<host-name>:<port>/SERVICE2/Resource5/Element5/<Status> | Positive response is received from Server. |
| **Step 7** | [REST Client Application]<br>Send Request to get details of Resourc6<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE2/Resource6<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 8** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status> | Positive response is received from Server. |

▽

△

| | | |
|---|---|---|
| **Step 9** | [REST Client Application]<br>Send Request to create Resorce6/Element7<br>Method: POST<br>URI: http://<host-name>:<port>/SERVICE2/Resource6/Element7<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 10** | [REST Server Application]<br>Server Response: HTTP/1.1 201 Created<br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7 | Positive response is received from Server. |
| **Step 11** | [REST Client Application]<br>Send Request to get details of Resource6<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE2/Resource6<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 12** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status><br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7/<Status> | Positive response is received from Server. |
| **Step 13** | [REST Client Application]<br>Send Request to delete [Element7]<br>Method: DELETE<br>URI: http://<host-name>:<port>/SERVICE2/Resource6/Element7<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 14** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element7 | Positive response is received from Server. |
| **Step 15** | [REST Client Application]<br>Send Request to get details of Resource6<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE2/Resource6<br>▽ | |

▽

△

| | | |
|---|---|---|
| | △<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 16** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>URI : http://<host-name>:<port>/SERVICE2/Resource6/Element6/<Status> | Positive response is received from Server. |
| **Step 17** | [REST Client Application]<br>Send an invalid URI Request<br>Method = GET,<br>URI: http://<host-name>:<port>/SERVICE5 | |
| **Step 18** | [REST Server Application]<br>Server replies with Status: 404<br>URI: http://<host-name>:<port>/SERVICE5 | Negative response is received from Server. |
| **Step 19** | [REST Client Application]<br>Send multiple requests from client.<br>Method = GET,<br>URI: http://<host-name>:<port>/SERVICE1/Resource1<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1<br>Method = GET,<br>URI: http://<host-name>:<port>/SERVICE1/Resource2<br>Host: <host-name><br>ContentLength : <length><br>ContentType: <application/json><br>Version: HTTP/1.1 | |
| **Step 20** | [REST Server Application]<br>Server replies with Status: 200 OK<br>URI: http://<hostname>:<port>/SERVICE1/Resource1/<status><br>Server replies with Status: 200 OK<br>URI: http://<hostname>:<port>/SERVICE1/Resource2/<status> | Positive 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 | **State** | Draft |

▽

△

| | | |
|---|---|---|
| **Affected Functional Cluster** | REST | |
| **Trace to RS Criteria** | [RS_CM_00301] | |
| **Reference to Test Environment** | STC_REST_00002 | |
| **Configuration Parameters** | RESTful API is configured | |
| **Summary** | - 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.<br>- Request is sent and response is received from server.<br>- Server application [APP2] provides a service [SERVICE2] with resource [Resource1] service [SERVICE2] is requested by [APP1] by HTTP and inter Process Communication (IPC).<br>- Response from server is processed and then client unsubscribe from the event.<br>- 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** | - [REST Tester] is connected to ECU.<br>- 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 | **State** | Draft |
| **Affected Functional Cluster** | REST | | |
| **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** | - Client and Server Applications communicates as per RESTful communication.<br>- First client is set up and request is created with URI and Methods<br>(GET/PUT/ POST/DELETE/OPTIONS).<br>- Request is sent and response is received from server as Object Graph having payload format JSON or XML.<br>- Response from server is processed and then client unsubscribe from the event.<br>- Client is stopped. | |
|---|---|---|
| **Pre-conditions** | - [REST Tester] is connected to ECU.<br>- 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]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength: <length><br>Accept: <application/json><br>Version: HTTP/1.1 | |
| **Step 2** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Content-Type: <application/json><br>Status: <Status1><br>URI : http://<host-name>:<port>/SERVICE1//Resource1/Element1/<Status> | Positive response is received from Server. |
| **Step 3** | [REST Client Application]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength: <length><br>Accept: <application/xml><br>Version: HTTP/1.1 | |
| **Step 4** | [REST Server Application]<br>Server Response: HTTP/1.1 200 OK<br>Content-Type: <application/xml><br>Status: <Status1><br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/<Status><br><resources><br><resource>Resource1</resource><br>▽ | Positive response is received from Server. |

▽

△

| | | |
|---|---|---|
| | △<br><elements><br><status>Status1</status><br></elements><br><elements><br><status>Status2</status><br></elements><br></resources> | |
| **Step 5** | [REST Client Application]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength: <length><br>Accept: <application/json><br>Version: HTTP/1.1 | |
| **Step 6** | [REST Server Application]<br>Server Response: HTTP/1.1<br>Content-Type: <application/xml><br>Status: <Status1><br>URI : http://<host-name>:<port>/SERVICE1//Resource1/Element1/<Status> | Response is rejected due to mismatch in Content type. |
| **Step 7** | [REST Client Application]<br>Send Request to get status of Resource1/Element1<br>Method: GET<br>URI: http://<host-name>:<port>/SERVICE1/Resource1/Element1/?Status<br>Host: <host-name><br>ContentLength: <length><br>Accept: <application/xml><br>Version: HTTP/1.1 | |
| **Step 8** | [REST Server Application]<br>Server Response: HTTP/1.1<br>Content-Type: <application/json><br>Status: <Status1><br>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 | **State** | Draft |
| **Affected Functional Cluster** | REST | | |
| **Trace to RS Criteria** | [RS_CM_00314] | | |
| **Reference to Test Environment** | STC_REST_00001 | | |
| **Configuration Parameters** | RESTful API is configured | | |
| **Summary** | - Client sends a handshake request to server to establish Websocket connection.<br>- Server returns a Websocket handshake response.<br>- Once the connection is established both client and server can listen for events.<br>- Event subscription message is sent as JSON over Websocket channel.<br>- Then Event cancellation message is sent as JSON over Websocket channel<br>- Response from server is processed and then client unsubscribe from the event.<br>- Client is stopped. | | |
| **Pre-conditions** | - [REST Tester] is connected to ECU.<br>- 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 handshake request to server to establish Websocket connection.<br>GET /<protocol> HTTP/1.1<br>Host: server.<URL><br>Upgrade: websocket<br>Connection: Upgrade<br>Sec-WebSocket-Key: <key>==<br>Origin: http://<URL><br>Sec-WebSocket-Protocol: <protocol><br>Sec-WebSocket-Version: <version> | | |
| **Step 2** | [REST Server Application]<br>Handshake from the server:<br>HTTP/1.1 101 Switching Protocols<br>Upgrade: websocket<br>Connection: Upgrade<br>Sec-WebSocket-Accept: <key>o=<br>Sec-WebSocket-Protocol: <protocol> | | Positive Handshake Response is received from Server. |
| **Step 3** | Websocket channel is opened during the first Event subscription and subscription message is sent as JSON over Websocket channel.<br>"type": "subscribe" | | |

▽

△

| | | |
|---|---|---|
| **Step 4** | [REST Server Application]<br><br>Server Responses to subscription message | Positive Subscription Response is received from Server. |
| **Step 5** | [REST Client Application] Event cancellation message is sent as JSON over Websocket channel<br><br>"type": "unsubscribe" | |
| **Step 6** | [REST Server Application]<br><br>Server Responses to cancellation message | Positive cancellation Response is received from Server. |
| **Step 7** | [REST Client Application] Request Error message from server. | |
| **Step 8** | [REST Server Application]<br><br>Server sends the Event error messages as JSON over the Websocket channel.<br><br>"type": "error" | Error message is received from Server. |

# 5 Test configuration and test steps for Execution Management

## 5.1 Test System

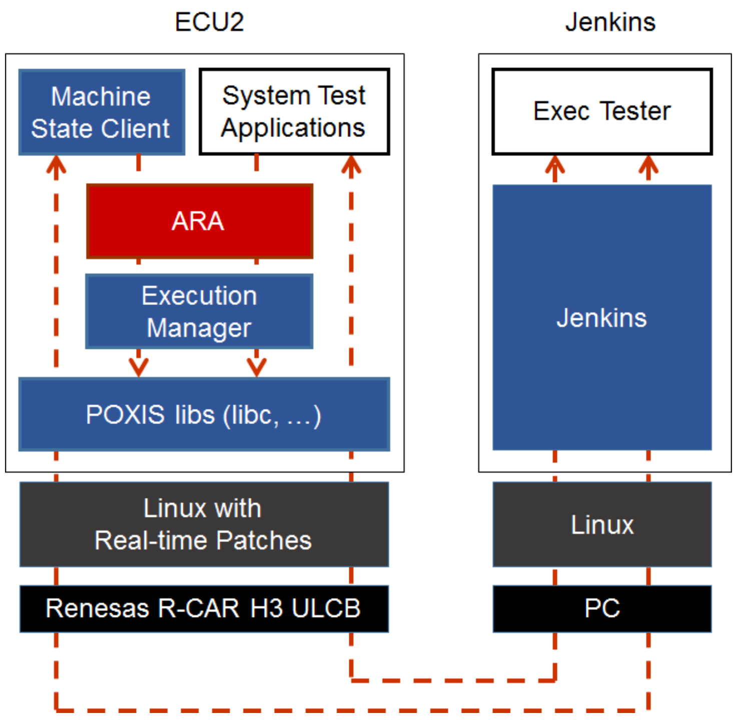

**Figure 5.1: Illustration of test setup for Execution Management.**

### 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** | *Startup (Initial Mode)* |
| | *Shutdown* |
| | *Restart* |
| | *Driving* |
| | *Parking* |

##### 5.1.1.1.2 Application Manifest

| **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* |

#### 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** | *Startup (Initial Mode)* |
| | *Shutdown* |
| | *Restart* |
| | *Driving* |
| | *Parking* |
| | |
| **Function Groups** | |
| **FG1** | *Off* |
| | *Running* |
| | *Fallback* |
| | *Diag* |
| | |
| **FG2** | *Off* |
| | *On* |
| | *Activate* |

#### 5.1.1.2.2 Application Manifest

| | | | |
|---|---|---|---|
| **Application Name** | **APP2** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | | |
| **Application Name** | **APP3** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | executionDependency | [APP2].*Running* |
| | | | |
| **Application Name** | **APP4** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | executionDependency | [APP3].*Running* |
| | | | |
| **Application Name** | **APP5** | | |
| Process | ModeDependentStartupConfig | functionGroup | [FG2].*On* and [FG2].*Activate* |
| **Application Name** | **APP6** | | |
| Process | ModeDependentStartupConfig | functionGroup | [FG2].*Activate* |

### 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** | *Startup (Initial Mode)* |
| | *Shutdown* |
| | *Restart* |
| | *Driving* |
| | *Parking* |

##### 5.1.1.3.2 Application Manifest

| | | | |
|---|---|---|---|
| **Application Name** | **APP2** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | | |
| **Application Name** | **APP3** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | | |
| **Application Name** | **APP4** | | |
| Process | ModeDependentStartupConfig | machineMode | *Parking* |
| | | | |
| **Application Name** | **APP5** | | |
| Process | ModeDependentStartupConfig | machineMode | *Parking* |

##### 5.1.1.3.3 ProcessToMachineMapping

| | | | | |
|---|---|---|---|---|
| **Application Name** | **APP2** | | | |
| Process | shallRunOn | ProcessorCore | CoreId | 1 and 2 |
| | | | | |
| **Application Name** | **APP3** | | | |
| Process | shallRunOn | ProcessorCore | CoreId | 1 and 2 |
| | | | | |
| **Application Name** | **APP4** | | | |
| Process | shallRunOn | ProcessorCore | CoreId | 3 and 4 |
| | | | | |
| **Application Name** | **APP5** | | | |
| Process | shallRunOn | ProcessorCore | CoreId | 3 and 4 |

#### 5.1.1.4 STC_EMO_00004

| | |
|---|---|
| **Configuration ID** | STC_EMO_00004 |
| **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] and [APP4].

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.

##### 5.1.1.4.1 Machine Manifest

| | |
|---|---|
| **Machine States** | *Startup (Initial Mode)* |
| | *Shutdown* |
| | *Restart* |
| | *Driving* |
| | *Parking* |
| | |
| **Function Groups** | |
| **FG1** | *Off* |
| | *On* |
| | *Activate* |

##### 5.1.1.4.2 Application Manifest

| | | | |
|---|---|---|---|
| **Application Name** | **APP2** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | schedulingPolicy | *schedulingPolicyRoundRobin* |
| | | schedulingPriority | 3 |
| | | | |
| **Application Name** | **APP3** | | |
| Process | ModeDependentStartupConfig | machineMode | *Driving* |
| | | executionDependency | [APP2].*Running* |
| | | schedulingPolicy | *schedulingPolicyOther* |
| | | schedulingPriority | 0 |

▽

△

| | | | |
|---|---|---|---|
| **Application Name** | **APP4** | | |
| Process | ModeDependentStartupConfig | functionGroup | [FG1].*On* |
| | | schedulingPolicy | *schedulingPolicyFifo* |
| | | schedulingPriority | 4 |
| | | | |
| **Application Name** | **APP5** | | |
| Process1 | ModeDependentStartupConfig | functionGroup | [FG1].*On* |
| | | schedulingPolicy | *schedulingPolicyRoundRobin* |
| | | schedulingPriority | 1 |
| | | startupConfig | environmentVariable<br>Key : APP_PATH<br>Value : /home/user1 |
| | | | startupOption<br>optionArgument : inputfile_1<br>CommandLineOptionKindEnum : commandLineLongForm<br>optionName : filename |
| Process2 | ModeDependentStartupConfig | functionGroup | [FG2].*On* |
| | | schedulingPolicy | *schedulingPolicyFifo* |
| | | schedulingPriority | 2 |
| | | startupConfig | environmentVariable<br>Key : APP_PATH<br>Value : /home/user2 |
| | | | startupOption<br>optionArgument : inputfile_2<br>CommandLineOptionKindEnum : commandLineLongForm<br>optionName : filename |

#### 5.1.1.4.3 Process Configuration

| **Process Name** | **Executable Reference** |
|---|---|
| *APP2Process* | *APP2Exec* |
| *APP3Process* | *APP3Exec* |
| *APP4Process* | *APP4Exec* |
| *APP5Process1* | *APP5Exec* |
| *APP5Process2* | *APP5Exec* |

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00100], [RS_EM_00101] | | |
| **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*.<br>A change of Machine State from *Startup* to *Parking* is requested and it is verified that [APP2], [APP3], [APP4] and [APP5] are not started.<br>A change of Machine State from *Parking* 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.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- 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]<br>Request change of Machine State to *Parking* for ECU2. | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Parking* from Execution Manager. | Machine State for ECU2 is changed to *Parking*. |
| **Step 3** | [Exec Tester]<br>Query execution status of [APP2]. | [APP2] is not executed. |
| **Step 4** | [Exec Tester]<br>Query execution status of [APP3]. | [APP3] is not executed. |
| **Step 5** | [Exec Tester]<br>Query execution status of [APP4]. | [APP4] is not executed. |
| **Step 6** | [Exec Tester]<br>Query execution status of [APP5]. | [APP5] is not executed. |
| **Step 7** | [Exec Tester]<br>Request change of Machine State to *Driving* for ECU2. | |
| **Step 8** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | Machine State for ECU2 is changed to *Driving*. |

▽

△

| | | |
|---|---|---|
| **Step 9** | [Exec Tester]<br>Query execution status of [APP2]. | [APP2] is executed. |
| **Step 10** | [Exec Tester]<br>Query execution status of [APP3]. | [APP3] is executed. |
| **Step 11** | [Exec Tester]<br>Query execution status of [APP4]. | [APP4] is executed. |
| **Step 12** | [Exec Tester]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00100], [RS_EM_00101] | | |
| **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.<br>- ECU ID for ECU2 is set to ECU2<br>- [APP2] has LT Application ID APPID2.<br>- Context ID for [APP2] is set to CTX2<br>- [APP3] has LT Application ID APPID3.<br>- Context ID for [APP3] is set to CTX3<br>- [APP4] has LT Application ID APPID4.<br>- Context ID for [APP4] is set to CTX4<br>- [APP5] has LT Application ID APPID5.<br>- Context ID for [APP5] is set to CTX5 | | |
| **Summary** | 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.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State Driving.<br>- Operating system on ECU2 has booted.<br>- 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]<br><br>Request change of Machine State to *Shutdown* for ECU2. | |
| **Step 2** | [SM]<br><br>Request for change of Machine State to *Shutdown* from Execution Manager. | Machine State for ECU2 is changed to *Shutdown*. |
| **Step 3** | [Exec Tester]<br><br>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<br><br>Message with context ID CTX3 and application ID APPID3 is received which is logged at [APP3] application termination<br><br>Message with context ID CTX4 and application ID APPID4 is received which is logged at [APP4] application termination<br><br>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 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00100], [RS_EM_00101], [RS_EM_00103] | | |
| **Reference to Test Environment** | STC_EMO_00002 | | |
| **Configuration Parameters** | - Machine State *Driving*, 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.<br><br>- ECU ID for ECU2 is set to *ECU2*<br><br>- [APP2] has LT Application ID *APPID2*<br><br>- Context ID for [APP2] is set to *CTX2*<br><br>- [APP3] has LT Application ID *APPID3*<br><br>▽ | | |

▽

△

| | | |
|---|---|---|
| | △<br>- Context ID for [APP3] is set to *CTX3*<br>- [APP4] has LT Application ID *APPID4*<br>- Context ID for [APP4] is set to *CTX4*<br>- [APP5] has LT Application ID *APPID5*<br>- Context ID for [APP5] is set to *CTX5*<br>- [APP6] has LT Application ID *APPID6*<br>- Context ID for [APP6] is set to *CTX6* | |
| **Summary** | When initialized the system state is *Startup*.<br>A change of Machine State from *Startup* to *Driving* 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.<br>A change of Machine State from *Driving* to *Parking* 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. | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- Function Group State for [FG2] is *Off*.<br>- 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]<br>Request change of Machine State to *Driving* for ECU2. | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | Machine State for ECU2 is changed to *Driving*. |
| **Step 3** | [Exec Tester]<br>Observe the log for applications [APP2] | Message with context ID *CTX2* and application ID *APPID2* is received which is logged at [APP2] application startup |
| **Step 4** | [Exec Tester]<br>Observe the log for applications [APP3] | Message with context ID *CTX3* and application ID *APPID3* is received which is logged at [APP3] application startup |
| **Step 5** | [Exec Tester]<br>Observe the log for applications [APP4] | Message with context ID *CTX4* and application ID *APPID4* is received which is logged at [APP4] application startup |
| **Step 6** | [Exec Tester]<br>Request change of Machine State to *Shutdown* for ECU2. | |
| **Step 7** | [SM]<br>Request for change of Machine State to *Parking* from Execution Manager. | Machine State for ECU2 is changed to *Parking*. |
| **Step 8** | [Exec Tester]<br>Observe the log for applications [APP4] | Message with context ID *CTX4* and application ID *APPID4* is received which is logged at [APP4] application termination |

▽

△

| | | |
|---|---|---|
| **Step 9** | [Exec Tester]<br><br>Observe the log for applications [APP3] | Message with context ID *CTX3* and application ID *APPID3* is received which is logged at [APP3] application termination |
| **Step 10** | [Exec Tester]<br><br>Observe the log for applications [APP2] | Message with context ID *CTX2* and application ID *APPID2* 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 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00100], [RS_EM_00101] | | |
| **Reference to Test Environment** | STC_EMO_00002 | | |
| **Configuration Parameters** | - Function Group State *Activate* and Function Group State *On* of [FG2] in which System Test Application [APP5] shall start is defined.<br><br>- Function Group State *Activate* of [FG2] in which System Test Application [APP6] shall start is defined | | |
| **Summary** | When initialized the Function Group State of [FG2] is *Off*.<br><br>A change of Function Group State of [FG2] to *On* is requested and the startup of the application [APP5] associated with this Function Group State is verified.<br><br>A change of Function Group State of [FG2] to *Activate* is requested and the startup of [APP6] associated with this Function Group State is verified. | | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br><br>- Software components on ECU2 are initialized.<br><br>- Function Group State [FG2] is *Off*.<br><br>- 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]<br><br>Request change of Function Group State [FG2] to *On*. | | |
| **Step 2** | [SM]<br><br>Request for change of Function Group State [FG2] to *On* from Execution Manager. | | Function Group State [FG2] for ECU2 is changed to *On*. |
| **Step 3** | [Exec Tester]<br><br>Query execution status of [APP5]. | | [APP5] is executed. |

▽

△

| | | |
|---|---|---|
| **Step 4** | [Exec Tester]<br>Request change of Function Group State [FG2] to *Activate*. | |
| **Step 5** | [SM]<br>Request for change of Function Group State [FG2] to *Activate* from Execution Manager. | Function Group State [FG2] for ECU2 is changed to *Activate*. |
| **Step 6** | [Exec Tester]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00009] | | |
| **Reference to Test Environment** | STC_EMO_00003 | | |
| **Configuration Parameters** | - 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.<br>- Each of the Applications [APP2], [APP3], [APP4] and [APP5] have one Executable invoked by a Process | | |
| **Summary** | A change of Machine State from *Startup* to *Driving* is requested. Start of [APP2] and [APP3] Processes from Execution Manager is checked.<br>Create or fork a Process from [APP2] Process and verify that no child Processes are created from [APP2] Process.<br>Execute [APP5] Process from [APP3] Process and verify that the [APP5] Process is not invoked from [APP3] Process. | | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- 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]<br>Request change of Machine State to Driving for ECU2. | | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | | Machine State for ECU2 is changed to *Driving*. |

▽

△

| | | |
|---|---|---|
| **Step 3** | Query execution status of [APP2] | [APP2] Process is executed |
| **Step 4** | [APP2]<br>Fork or create a Process from [APP2] | |
| **Step 5** | [Exec Tester]<br>Get the Process ID of the Execution Manager | Received the Process ID of Execution Manager.<br>*EXMPID*(1) |
| **Step 6** | [Exec Tester]<br>Get the Process ID of [APP2] Process | Received the Process ID of [APP2] Process<br>*APP2PID* |
| **Step 7** | [Exec Tester]<br>Get the Parent Process ID of [APP2] Process | The Parent Process ID of [APP2] Process is received as *EXMPID*(1) |
| **Step 8** | [Exec Tester]<br>Get the Child Processes of Process ID *APP2PID* | No child Processes of [APP2] Process shall be received. |
| **Step 9** | Query execution status of [APP3] | [APP3] Process is executed |
| **Step 10** | [APP3]<br>Execute or Invoke [APP5] Process from [APP3] Process | [APP5] Process is not executed |

### 5.2.6 [STS_EMO_00006] Execution Management shall create one POSIX process for each Executable instance and shall launch the process with the scheduling policy and priority configured in the Execution Manifest

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that the one POSIX process is created for each Executable instance configured and the scheduling policy and priority for the process is assigned as specified in the Execution Manifest. | | |
| **ID** | STS_EMO_00006 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00002] | | |
| **Reference to Test Environment** | STC_EMO_00004 | | |
| **Configuration Parameters** | - Machine State Driving, in which Processes [APP2].Process and [APP3].Process shall start is defined with [APP3].Process having dependency on [APP2].Process<br>The scheduling policy and scheduling priority are configured as schedulingPolicyRoundRobin and 3 respectively for [APP2].Process and schedulingPolicyOther and 0 respectively for [APP3].Process<br>- Function Group State On of [FG2] in which Process [APP4].Process shall start is defined with scheduling policy as schedulingPolicyFifo and scheduling priority 4. | | |

▽

△

| | | |
|---|---|---|
| **Summary** | A change of Machine State from Startup to Driving is requested.<br>Start of [APP2].Process from the Execution Manager with the configured scheduling policy (schedulingPolicyRoundRobin) and priority (3) is checked. Start of [APP3].Process from the Execution Manager with the configured scheduling policy (schedulingPolicyOther) and priority (0) is checked after the start of [APP2].Process, since [APP3].Process has dependency on [APP2].Process<br>A change of Function Group State of [FG1] to On is requested and the startup of the Process [APP4].Process is verified with the configured scheduling policy (schedulingPolicyFifo) and scheduling priority (4). | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- ECU2 Function Group State [FG2] is *Off*.<br>- 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]<br>Request change of Machine State to Driving for ECU2. | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | Machine State for ECU2 is changed to *Driving*. |
| **Step 3** | [Exec Tester]<br>Query execution status of [APP2] Process | [APP2] Process is executed |
| **Step 4** | [Exec Tester]<br>Get the Process ID of the Execution Manager | Received the Process ID of Execution Manager.<br>*EXMPID*(1) |
| **Step 5** | [Exec Tester]<br>Get the Process ID of the [APP2] Process | Received the Process ID of [APP2] Process.<br>*APP2PID* |
| **Step 6** | [Exec Tester]<br>Get the Parent Process ID of [APP2] | The Parent Process ID of [APP2] is received as *EXMPID* |
| **Step 7** | [Exec Tester]<br>Get the scheduling policy of [APP2] Process | Scheduling policy is received as SCHED_RR |
| **Step 8** | [Exec Tester]<br>Get the scheduling priority of [APP2] Process | Scheduling priority is received as 3 |
| **Step 9** | [Exec Tester]<br>Get the Process ID of the [APP3] Process | Received the Process ID of [APP3] Process.<br>*APP3PID* |
| **Step 10** | [Exec Tester]<br>Get the Parent Process ID of [APP3] | The Parent Process ID of [APP3] is received as *EXMPID* |
| **Step 11** | [Exec Tester]<br>Get the scheduling policy of [APP3] Process | Scheduling policy is received as SCHED_OTHER |
| **Step 12** | [Exec Tester]<br>Get the scheduling priority of [APP2] Process | Scheduling priority is received as 0 |
| **Step 13** | [SM]<br>Request change of Function Group State [FG2] to *On*. | |

▽

△

| | | |
|---|---|---|
| **Step 14** | [Exec Tester]<br><br>Request for change of Function Group State [FG2] to *On* from Execution Manager. | Function Group State [FG2] for ECU2 is changed to *On*. |
| **Step 15** | [Exec Tester]<br><br>Get the Process ID of the [APP4] Process | Received the Process ID of [APP4] Process.<br><br>*APP4PID* |
| **Step 16** | [Exec Tester]<br><br>Get the Parent Process ID of [APP4] | The Parent Process ID of [APP4] is received as *EXMPID* |
| **Step 17** | [Exec Tester]<br><br>Get the scheduling policy of [APP4] Process | Scheduling policy is received as SCHED_FIFO |
| **Step 18** | [Exec Tester]<br><br>Get the scheduling priority of [APP4] Process | Scheduling priority is received as 4 |

### 5.2.7 [STS_EMO_00007] Execution Management shall support multiple instantiation of Executable with different startup parameters from different Processes

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that Execution Management shall support multiple instantiation of Executable from different POSIX processes with different startup parameters. | | |
| **ID** | STS_EMO_00007 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |
| **Trace to RS Criteria** | [RS_EM_00010] | | |
| **Reference to Test Environment** | STC_EMO_00004 | | |
| **Configuration Parameters** | Function Group State *On* of [FG1] in which Process [APP5].Process1 shall start is defined with following StartupConfig<br><br>schedulingPolicy : schedulingPolicyRoundRobin<br><br>schedulingPriority : 1<br><br>StartupOption : filename = inputfile_1<br><br>Environment Variable : APP_PATH = /home/user1<br><br>Function Group State *On* of [FG1] in which Process [APP5].Process2 shall start is defined with following StartupConfig<br><br>schedulingPolicy : schedulingPolicyFifo<br><br>schedulingPriority : 2<br><br>StartupOption : filename = inputfile_2<br><br>Environment Variable : APP_PATH = /home/user2 | | |

▽

△

| | | |
|---|---|---|
| **Summary** | A change of Function Group State of [FG1] to *On* is requested. startup of the Process [APP5].Process1 is verified<br><br>A change of Function Group State of [FG2] to *On* is requested. startup of the Process [APP5].Process2 is verified<br><br>It is verified that the same Executable *APP5Exec* is invoked from both the Processes [APP5].Process1 and [APP5].Process2 with different startup parameters as specified below:<br><br>[APP5].Process1<br><br>scheduling policy : schedulingPolicyRoundRobin<br><br>scheduling priority : 1<br><br>argument : filename = inputfile_1<br><br>environment variable : APP_PATH = /home/user1<br><br>[APP5].Process2<br><br>scheduling policy : schedulingPolicyFifo<br><br>scheduling priority : 2<br><br>argument : filename = inputfile_2<br><br>environment variable : APP_PATH = /home/user2<br><br>Note: *APP5Exec* shall invoke a main program with 3 arguments which specifies argument count, argument list and environment list. | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br><br>- Software components on ECU2 are initialized.<br><br>- ECU2 is in Machine State *Startup*.<br><br>- ECU2 Function Group State [FG2] is *Off*.<br><br>- 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]<br><br>Request change of Function Group State [FG1] to *On*. | |
| **Step 2** | [SM]<br><br>Request for change of Function Group State [FG1] to *On* from Execution Manager | Function Group State [FG1] for ECU2 is changed to *On*. |
| **Step 3** | [Exec Tester]<br><br>Query execution status of [APP5].Process1 | [APP5].Process1 is executed |
| **Step 4** | [Exec Tester]<br><br>Get the Process ID of the [APP5].Process1 | Received the Process ID of [APP5].Process1<br><br>*APP5PID1* |
| **Step 5** | [Exec Tester]<br><br>Get the scheduling policy of [APP5].Process1 | Scheduling policy is received as SCHED_RR |
| **Step 6** | [Exec Tester]<br><br>Get the scheduling priority of [APP5].Process1 | Scheduling priority is received as 1 |
| **Step 7** | [APP5].Process1<br><br>Read the arguments | |

▽

△

| | | |
|---|---|---|
| **Step 8** | [Exec Tester]<br>Get the arguments of [APP5].Process1 | Check if only one argument is received and the argument received is<br>filename = inputfile_1 |
| **Step 9** | [APP5].Process1<br>Read the environment variables | |
| **Step 10** | [Exec Tester]<br>Get the environment variables of [APP5].Process1 | Check if the environment variable APP_PATH has /home/user1 |
| **Step 11** | [Exec Tester]<br>Request change of Function Group State [FG2] to *On*. | |
| **Step 12** | [SM]<br>Request for change of Function Group State [FG2] to *On* from Execution Manager | Function Group State [FG2] for ECU2 is changed to *On*. |
| **Step 13** | [Exec Tester]<br>Query execution status of [APP5].Process2 | [APP5].Process2 is executed |
| **Step 14** | [Exec Tester]<br>Get the Process ID of the [APP5].Process2 | Received the Process ID of [APP5].Process2<br>*APP5PID2* |
| **Step 15** | [Exec Tester]<br>Get the scheduling policy of [APP5].Process2 | Scheduling policy is received as SCHED_FIFO |
| **Step 16** | [Exec Tester]<br>Get the scheduling priority of [APP5].Process2 | Scheduling priority is received as 2 |
| **Step 17** | [APP5].Process2<br>Read the arguments | |
| **Step 18** | [Exec Tester]<br>Get the arguments of [APP5].Process2 | Check if only one argument is received and the argument received is<br>filename = inputfile_2 |
| **Step 19** | [APP5].Process1<br>Read the environment variables | |
| **Step 20** | [Exec Tester]<br>Get the environment variables of [APP5].Process2 | Check if the environment variable APP_PATH has /home/user2 |

### 5.2.8 [STS_EMO_00008] Execution Management shall support self initiated graceful shutdown of Processes

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that Execution Management shall support self initiated graceful shutdown of processes. | | |
| **ID** | STS_EMO_00008 | **State** | Draft |
| **Affected Functional Cluster** | Execution Management | | |

▽

△

| | | |
|---|---|---|
| **Trace to RS Criteria** | [RS_EM_00011] | |
| **Reference to Test Environment** | STC_EMO_00003 | |
| **Configuration Parameters** | Machine State Driving, in which all System Test Applications [APP2] shall start is defined | |
| **Summary** | A change of Machine State from Startup to Driving is requested. Start of [APP2] Process is checked.<br>Initiate self termination from [APP2] Process and check that Execution Manager supports the self initiated shutdown of Process | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- 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]<br>Request change of Machine State to *Driving* for ECU2. | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | Machine State for ECU2 is changed to *Driving*. |
| **Step 3** | [Exec Tester]<br>Query execution status of [APP2] Process | [APP2] Process is executed |
| **Step 4** | [Exec Tester]<br>Get the Process ID of the [APP2] Process1 | Received the Process ID of [APP2] Process<br>*APP2PID* |
| **Step 5** | [APP2] Process<br>Report kTerminating state using API ExecutionClient::ReportExecutionState to Execution Manager | |
| **Step 6** | [APP2] Process<br>Exit from [APP2] Process | |
| **Step 7** | [Exec Tester]<br>Get the list of currently running process | Check if *APP2PID* does not exist in the list of currently running process |

### 5.2.9 [STS_EMO_00009] Execution Management shall support binding of processes and its associated threads to specified set of cores

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that the Execution Management shall support the binding of processes and its associated threads to specific set of cores as specified in the Execution Manifest. | | |
| **ID** | STS_EMO_00009 | **State** | Draft |

▽

△

| | | |
|---|---|---|
| **Affected Functional Cluster** | Execution Management | |
| **Trace to RS Criteria** | [RS_EM_00008] | |
| **Reference to Test Environment** | STC_EMO_00003 | |
| **Configuration Parameters** | - Machine State Driving, in which all System Test Applications [APP2], [APP3], [APP4] and [APP5] shall start is defined<br>- [APP2].Process and [APP3].Process are mapped to cores 1 and 2<br>- [APP4].Process and [APP5].Process are mapped to cores 3 and 4 | |
| **Summary** | A change of Machine State from Startup to Driving is requested.<br>Start of [APP2] Process is checked. Also it is checked that [APP2] Process runs on core 1 and 2 as configured in the Execution Manifest.<br>Threads are created inside the [APP2] Process and it is checked that threads are running on core 1 or 2.<br>Assign core 1 to thread created inside [APP2] Process and it is checked that the thread runs in core 1.<br>Assign core 3 to thread created inside [APP2] Process and it is checked that the thread does not run in core 3, since core 3 is not set for [APP2] Process | |
| **Pre-conditions** | - Exec Tester is connected to ECU2 via TCP.<br>- Software components on ECU2 are initialized.<br>- ECU2 is in Machine State *Startup*.<br>- 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]<br>Request change of Machine State to *Driving* for ECU2. | |
| **Step 2** | [SM]<br>Request for change of Machine State to *Driving* from Execution Manager. | Machine State for ECU2 is changed to *Driving*. |
| **Step 3** | [Exec Tester]<br>Query execution status of [APP2] Process | [APP2] Process is executed |
| **Step 4** | [Exec Tester]<br>Get the Process ID of the [APP2] Process1 | Received the Process ID of [APP2] Process<br>*APP2PID* |
| **Step 5** | [Exec Tester]<br>Get the core in which [APP2] Process is running | Check if the [APP2] Process is running in core 1 or 2 |
| **Step 6** | [APP2] Process<br>Create a thread *APP2ProcThread1* inside the [APP2] Process | |
| **Step 7** | [Exec Tester]<br>Get the core in which the thread *APP2ProcThread1* is running | Check if the thread *APP2ProcThread1* is running in core 1 or 2 |
| **Step 8** | [APP2] Process<br>Assign core 1 to the thread *APP2ProcThread1* | |
| **Step 9** | [Exec Tester]<br>Get the core in which the thread *APP2ProcThread1* is running | Check if the thread *APP2ProcThread1* is running in core 1 |

▽

△

| | | |
|---|---|---|
| **Step 10** | [APP2] Process<br>Create a thread *APP2ProcThread2* inside the [APP2] Process | |
| **Step 11** | [Exec Tester]<br>Get the core in which the thread *APP2ProcThread2* is running | Check if the thread *APP2ProcThread2* is running in core 1 or 2 |
| **Step 12** | [APP2] Process<br>Assign core 3 to the thread *APP2ProcThread2* | |
| **Step 13** | [Exec Tester]<br>Get the core in which the thread *APP2ProcThread2* is running | Check if the thread *APP2ProcThread2* is running in core 1 or 2 |

# 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.

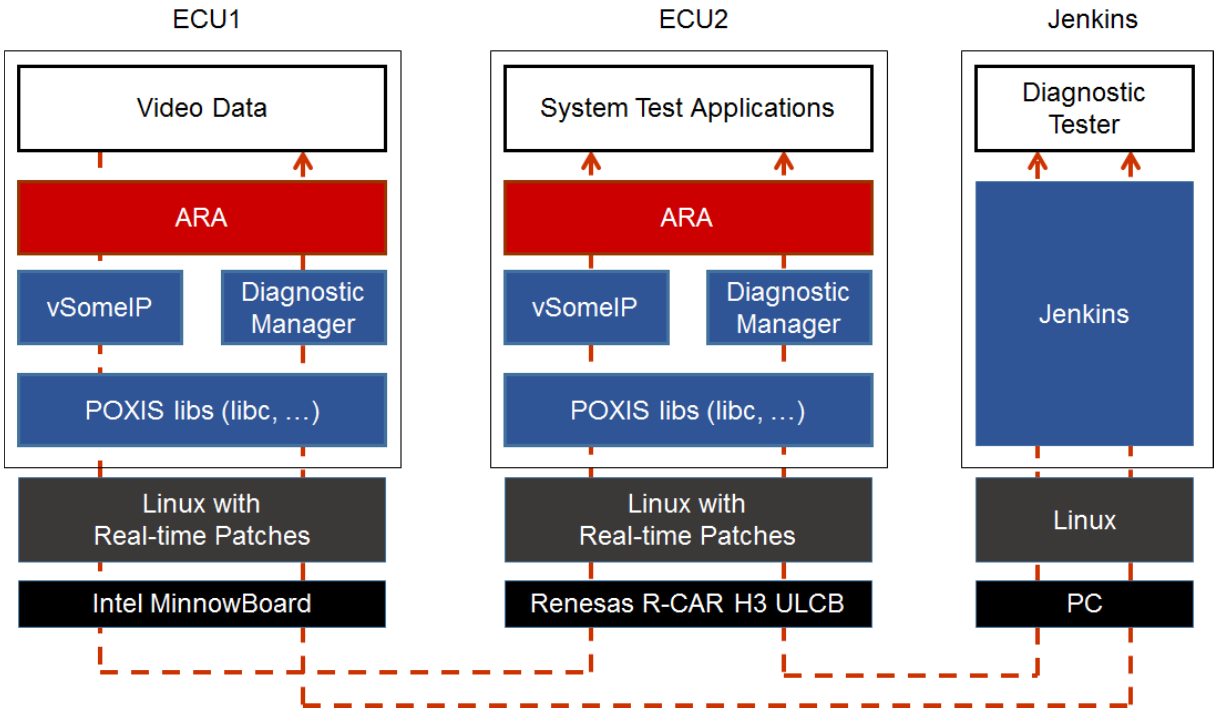

**Figure 6.1: Illustration of test setup for Diagnostics.**

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_DIAG_00001 | | |
| **Configuration Parameters** | - Diagnostics module:<br>• Service instance for service ReadDataByIdentifier with DID <0x0001> is configured.<br>• 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.<br>- 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]<br>Send UDS Request to query value of <int1>:<br>UDS-Service: ReadDataByIdentifier<br>UDS-Payload: 0x22 ... | | |
| **Step 2** | [APP1]<br>Start mechanism to read the value of <int1>. | | |
| **Step 3** | [Diagnostic Tester]<br>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]<br>Start mechanism to change the value of <int1> by <delta>. | | |
| **Step 5** | [Diagnostic Tester]<br>Send UDS Request to query value of <int1>:<br>UDS-Service: ReadDataByIdentifier<br>UDS-Payload: 0x22 ... | | |
| **Step 6** | [APP1]<br>Start mechanism to read value of <int1> and return it as DID data. | | |

▽

△

| | | |
|---|---|---|
| **Step 7** | [Diagnostic Tester]<br>Receive UDS response and save value of <int1> in <var2>. | Positive response received (0x62 ...).<br>Payload of UDS response contains DID data. Compare values of <var1> and <var2>. <var2> should be greater than <var1> by <delta> i.e.<br><var2>=<var1> + <delta>. |
| **Step 8** | [Diagnostic Tester]<br>Send UDS Request to query data with a non-implemented DID:<br>UDS-Service: ReadDataByIdentifier<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_DIAG_00001 | | |
| **Configuration Parameters** | - The following service is configured<br>[SERVICE1] in [APP1] - In this [SERVICE1], two different contents are available<br>• <Content1><br>• <Content2><br>- Diagnostics module:<br>• Service instance for service RoutineControl with RID <0x0001> is configured and only available in Extended Diagnostic Session.<br>• 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.<br>- Software components on [ECU1] are initialized.<br>- [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]<br>Send UDS request to change content of [SERVICE1]:<br>UDS-Service: RoutineControl<br>UDS-Payload: 0x31 0x01 ... | Negative response received: Service Not Supported in Active Session (0x7F 0x31 0x7F). |
| **Step 2** | [Diagnostic Tester]<br>Send UDS request to start an Extended Diagnostic Session:<br>UDS-Service: DiagnosticSessionControl<br>UDS-Payload: 0x10 0x03 | Positive response received (0x50 0x03). |
| **Step 3** | [Diagnostic Tester]<br>Send UDS request to change content of [SERVICE1] from <Content1> to <Content2>:<br>UDS-Service: RoutineControl<br>UDS-Payload: 0x31 0x01 ... | |
| **Step 4** | [APP1]<br>Start mechanism to change content of [SERVICE1] from <Content1> to <Content2> | Content of Service is changed to <Content2> |
| **Step 5** | [APP1]<br>Return from Subfunction Start of Routine with RID <0x0001>. | |
| **Step 6** | [Diagnostic Tester]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_DIAG_00001 | | |
| **Configuration Parameters** | Diagnostics module:<br>• Service instance for service RoutineControl with RID <0x0001> is configured and only available in Extended Diagnostic Session.<br>• Service Diagnostic Session Control and Extended Diagnostic Session time out is configured.<br>• 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.<br>- 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]<br>Send UDS request to start an Extended Diagnostic Session:<br>UDS-Service: DiagnosticSessionControl(SID 0x10)<br>UDS-Payload: 0x10 0x03 | Positive response received<br>(0x50 0x03). |
| **Step 2** | [Diagnostic Tester]<br>Wait for time <t1> such that <t1> is less than Diagnostic session timer timeout. | |
| **Step 3** | [Diagnostic Tester]<br>Send UDS request Tester Present with suppressPosRspMsg IndicationBit is set to FALSE.<br>UDS-Service: TesterPresent (SID 0x3E)<br>UDS-Payload: 0x3E 0x00 | Positive response received<br>(0x7E 0x00). |
| **Step 4** | [Diagnostic Tester]<br>Wait for time <t2> such that -<br>1) <t2> is greater than Diagnostic session timer timeout.<br>2) <t2> is less than sum of Extended session timer and Diagnostic session timer timeout. | |
| **Step 5** | [Diagnostic Tester]<br>Send UDS request RoutineControl to confirm if Extended Session is active.<br>UDS-Service: RoutineControl (SID 0x31)<br>UDS-Payload: 0x31 0x01 ... | Positive response received<br>(0x71 ...). |
| **Step 6** | [Diagnostic Tester]<br>Stop sending TesterPresent and wait for Extended Diagnostic Session to time out | |
| **Step 7** | [Diagnostic Tester]<br>Send UDS request RoutineControl to confirm if Extended Session is active.<br>UDS-Service: RoutineControl<br>UDS-Payload: 0x31 0x01 ... | Negative response received: Service Not Supported in Active Session (0x7F 0x31 0x7F (NRC)). |
| **Step 8** | [Diagnostic Tester]<br>Send UDS request to start an Extended Diagnostic Session:<br>UDS-Service: DiagnosticSessionControl<br>UDS-Payload: 0x10 0x03 | Positive response received<br>(0x50 0x03). |
| **Step 9** | [Diagnostic Tester]<br>Wait for time <t1> such that <t1> is less than Diagnostic session timer timeout. | |

▽

△

| | | |
|---|---|---|
| **Step 10** | [Diagnostic Tester]<br>Send UDS request TesterPresent with suppressPosRspMsg IndicationBit is set to TRUE.<br>UDS-Service: TesterPresent<br>UDS-Payload: 0x3E 0x80 | No response received for UDS request TesterPresent. |
| **Step 11** | [Diagnostic Tester]<br>Wait for time <t2> such that -<br>1) <t2> is greater than Diagnostic session timer timeout.<br>2) <t2> is less than sum of Extended session timer and Diagnostic session timer timeout. | |
| **Step 12** | [Diagnostic Tester]<br>Send UDS request RoutineControl to confirm if Extended Session is active.<br>UDS-Service: RoutineControl<br>UDS-Payload: 0x31 ... | Positive response received<br>(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 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **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<br>- 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]<br>Send UDS Request to query value of <int1>:<br>UDS-Service: ReadDataByIdentifier<br>UDS-Payload: 0x22 ... | | |

▽

△

| | | |
|---|---|---|
| **Step 2** | [APP1]<br>Wait for invocation. | Implementation of method Read for DID <0x0001> is invoked. |
| **Step 3** | [Diagnostic Tester]<br>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]<br>Send UDS Request to overwrite value of <int1> with <int2><br>UDS-Service:<br>WriteDataByIdentifier<br>UDS-Payload: 0x2E ... | |
| **Step 5** | [Diagnostic Tester]<br>Receive UDS response. | Positive response received (0x6E ...) after successful write. |
| **Step 6** | [Diagnostic Tester]<br>Send UDS request to query value of <int1><br>UDS-Service:<br>ReadDataByIdentifier<br>UDS-Payload: 0x22 ... | |
| **Step 7** | [APP1]<br>Wait for invocation. | Implementation of method Read for DID <0x0001> is invoked. |
| **Step 8** | [Diagnostic Tester]<br>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]<br>Compare <var> and <int2> values. | Both values should be equal. |

### 6.2.5 [STS_DIAG_00005] Utilization of Diagnostic service InputOutputControlByIdentifier (0x2F) by External Tester via UDS messages over DoIP.

| | | | |
|---|---|---|---|
| **Test Objective** | Verification of correct behavior of Diagnostic service InputOutputControlByIdentifier (0x2F) by External Tester via UDS messages over DoIP. | | |
| **ID** | STS_DIAG_00004 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **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 InputOutputControlByIdentifier with DID <0x0001> are configured. - Methods ShortTermAdjustment , FreezeCurrentState ,ReturnControlToECU ,ResettoDefault for InputOutputControlByIdentifier for DID <0x001>are available | | |
| **Summary** | This basic test tries to send request for ShortTermAdjustment/FreezeCurrentState/ResettoDefault/FreezeCurrentState for DID <0x001> contained by [APP1]on [ECU1] over the AP Diagnostics Module. This test tries to substitute values of the input for DID <0x0001> and verify the output as desired | | |

▽

△

| | | |
|---|---|---|
| **Pre-conditions** | - [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port<br>- 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]<br>Send UDS Request for ShortTermAdjustment to value <x> for DID <0x0001> SID :0x2F ,InputOutputcontrolParameter = 0x03(ShortTermAdjustment) Payload : 0x2F 0x00 0x01 03 ... | |
| **Step 2** | [APP1]<br>Wait for invocation. | Implementation of method ShortTermAdjustment for DID <0x0001> is invoked. |
| **Step 3** | [Diagnostic Tester]<br>Receive UDS response with desired ShortTermAdjustment | Positive response received (0x6F ...). Payload of UDS response contains DID data with desired shorttermadjustment. |
| **Step 4** | [Diagnostic Tester]<br>Send UDS Request to Freeze State of DID<0x001><br>SID :0x2F ,InputOutputcontrolParameter = 0x02(FreezeCurrentState) UDS-Payload: 0x2F ... | |
| **Step 5** | [APP1]<br>Wait for invocation. | Implementation of method FreezeCurrentState for DID <0x0001> is invoked. |
| **Step 6** | [Diagnostic Tester]<br>Receive UDS response with Current State Freezed. | Positive response received (0x6F ...). Payload of UDS response contains DID data . |
| **Step 7** | [Diagnostic Tester]<br>Send UDS request to ResetToDefault<br>SID :0x2F ,InputOutputcontrolParameter = 0x01(ResetToDefault)<br>UDS-Payload: 0x2F ... | |
| **Step 8** | [APP1]<br>Wait for invocation. | Implementation of method ResetToDefault for DID <0x0001> is invoked. |
| **Step 9** | [Diagnostic Tester]<br>Receive UDS response | Positive response received (0x6F ...). Payload of UDS response contains DID data reset to default . |

### 6.2.6 [STS_DIAG_00006] Utilization of Diagnostic service ClearDTC (0x14) by External Tester via UDS messages over DoIP.

| | | | |
|---|---|---|---|
| **Test Objective** | Verification of correct behavior of Diagnostic service ClearDTC (0x14) by External Tester via UDS messages over DoIP. | | |
| **ID** | STS_DIAG_00006 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |

▽

△

| | | |
|---|---|---|
| **Trace to RS Criteria** | RS traceability will be added in next release | |
| **Reference to Test Environment** | STC_DIAG_00001 | |
| **Configuration Parameters** | Diagnostics module:<br>- Service instances for service Clear DTC(0x14) are configured.<br>- GroupofDTC <gtc1> is configured. | |
| **Summary** | | |
| **Pre-conditions** | - [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port<br>- 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]<br>Send UDS request to clear GroupofDTC<gtc1> related to event <e1><br>SID :0x14<br>Payload : 0x14 0xFF FF | |
| **Step 2** | [APP1]<br>Implementation of Service Clear DTC is invoked. | Check if requested GroupofDTC<gtc1> is present in the configured group of DTC. If yes, Send response. |
| **Step 3** | [Diagnostic Tester]<br>Receive UDS response | Positive response received (0x54 ...). Payload of UDS response contains status of cleared DTC. |
| **Step 4** | [Diagnostic Tester]<br>Send UDS request to read cleared GroupofDTC<gtc1> related to event <e1><br>SID :0x19<br>Payload : 0x19 .. | |
| **Step 5** | [APP1]<br>Invoke implementation of Diagnostic Service Read DTC | Check if DTC is available. |
| **Step 6** | [Diagnostic Tester]<br>Receive UDS response | Positive response (0x59)with no available DTC is received |
| **Step 7** | [Diagnostic Tester]<br>Send UDS request to clear GroupofDTC<gtc1> related to event <e1><br>SID :0x14<br>Payload: 0x14 0xFF FF . | |
| **Step 8** | [APP1]<br>Implementation of service Clear DTC is invoked.Check Length of requested request | If length of requested UDS request is incorrect send NRC-13. |
| **Step 9** | [Diagnostic Tester]<br>Receive UDS response for Clear DTC. | Negative response received (0x7F 0x14 0x13...). |

▽

△

| | | |
|---|---|---|
| **Step 10** | [Diagnostic Tester]<br>Send UDS request to clear GroupofDTC<gtc1> related to event <e1><br>SID :0x14<br>Payload: 0x14 0xFF FF | |
| **Step 11** | [APP1]<br>Implementation of service Clear DTC is invoked.Check if requested DTC group is available. | Group of DTC is not available,Send NRC-31 . |
| **Step 12** | [Diagnostic Tester]<br>Receive UDS response | Negative response received (0x7F 0x14 0x31...) |

### 6.2.7 [STS_DIAG_00007] Utilization of Diagnostic service SecurityAccess (0x27) by External Tester via UDS messages over DoIP.

| | | | |
|---|---|---|---|
| **Test Objective** | Verification of correct behavior of Diagnostic service SecurityAccess (0x27) by External Tester via UDS messages over DoIP. | | |
| **ID** | STS_DIAG_00007 | **State** | Draft |
| **Affected Functional Cluster** | Diagnostic | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_DIAG_00001 | | |
| **Configuration Parameters** | Diagnostics module:<br>- Service instances for service Security access are configured<br>- Service instances for Service ReadDataByIdentifier with DID <0x0001> are configured.<br>- Sub functions (SecurityAccessType) are configured. | | |
| **Summary** | This basic test tries to get an access of an ECU using Diagnostic service Security Access and try to access some secured parameters (DID <0x0001>)of an ECU. Tester first request for SEED, ECU responds with the SEED Value(random 2 byte number). Tester then generates the Key using the received SEED(Lower nibble of each byte masked with 0 ,Note that this could be OEM specific we are considering this as an example for demonstration) and send it to an ECU.ECU then verifies the key and grants access (Positive Response) .If Length of the request /sub function is not supported, then ECU shall send NRC | | |
| **Pre-conditions** | - [Diagnostic Tester] is connected to [ECU1] via TCP socket on DoIP-Port<br>- 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]<br>Send UDS request to gain SecurityAccessType -1<br>SID : 0x27<br>Payload - 0x27 01 .. | | |

▽

△

| | | |
|---|---|---|
| **Step 2** | [APP1]<br>Implementation of Method - RequestSeed is invoked. | Check the length of the UDS security request, if the length is not correct send NRC-13 |
| **Step 3** | [Diagnostic Tester]<br>Receive UDS response | Negative response received (0x7F 0x27 0x13 ...) |
| **Step 4** | [Diagnostic Tester]<br>Send UDS request to gain SecurityAccessType - 2<br>SID : 0x27<br>Payload - 0x27 02 .. | |
| **Step 5** | [APP1]<br>Implementation of Method - RequestSeed is invoked. | Check if the sub function (SecurityAccessType -2) is supported or not. If not send NRC-12 |
| **Step 6** | [Diagnostic Tester]<br>Receive UDS response | Negative response (0x7F 0x27 0x12) |
| **Step 7** | [Diagnostic Tester]<br>Send UDS request to gain SecurityAccessType - 1<br>SID : 0x27<br>Payload - 0x27 01 .. | |
| **Step 8** | [APP1]<br>Implementation of method RequestSeed is invoked | Seed (2 bytes of random number)is generated successfully and response is sent |
| **Step 9** | [Diagnostic Tester]<br>Send Request to SendKey<br>SID: 0x27<br>Payload : 0x27 0x02 ... | |
| **Step 10** | [APP1]<br>Invoke method to verify received key | Check if the received Key is equal to internally generated key ,if yes send positive response |
| **Step 11** | [Diagnostic Tester]<br>Receive positive response. | Positive response (0x67 ..) is received |
| **Step 12** | [Diagnostic Tester]<br>Send Request to read a secured paramter <var1> using ReadDID Service<br>SID : 0x22<br>Payload : 0x22 0x00 0x01 | |
| **Step 13** | [APP1]<br>Invoke Service ReadDataByIdentifier | Provide value of <var1> as a response |
| **Step 14** | [DiagnosticTester]<br>Receive UDS Service response | Positive response (0x62 0x00 0x01 var1 ) |

# 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.

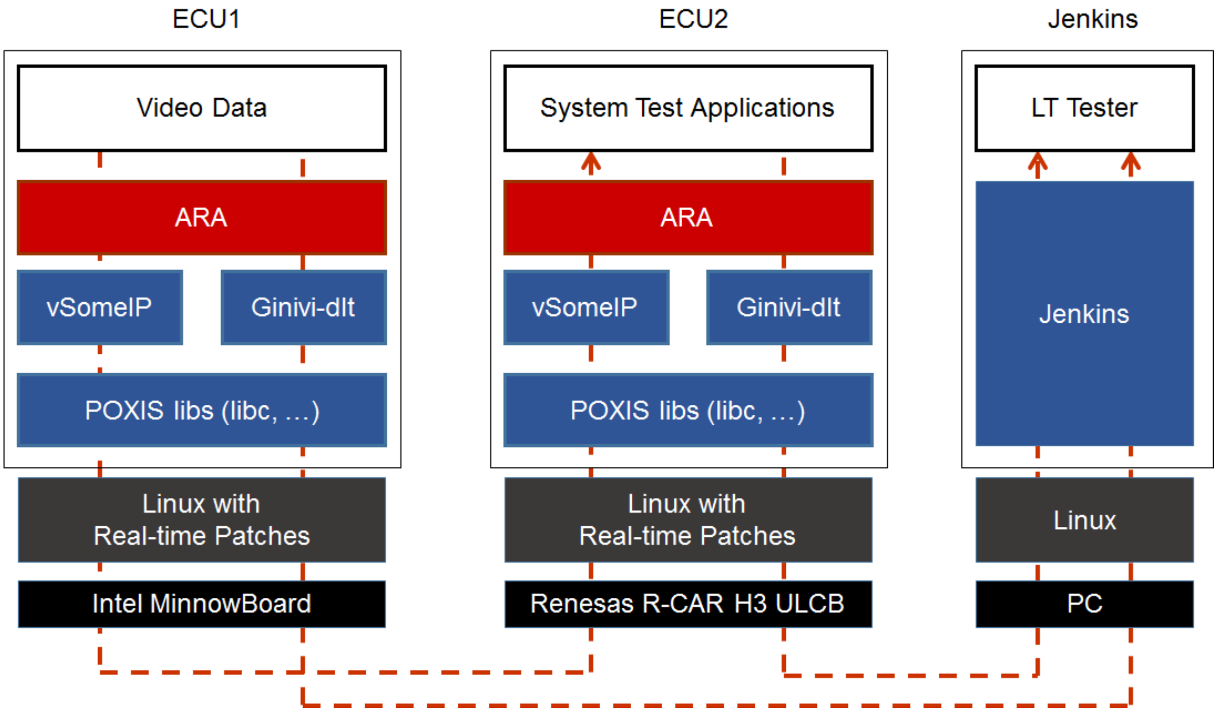

**Figure 7.1: Illustration of test setup for Logging and Tracing.**

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Logging and Tracing | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_LT_00001 | | |
| **Configuration Parameters** | - LT module in ECU1 is configured properly:<br>- ECU ID for ECU1 is set to ECU1<br>- [APP1] has LT Application ID APPID1.<br>- 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.<br>• Software components on [ECU1] are initialized.<br>• 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]<br>Receive log messages. | Tester receives log messages every 0.5 seconds.<br>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]<br>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]<br>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]<br>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]<br>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]<br>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]<br><br>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 | **State** | Draft |
| **Affected Functional Cluster** | Logging and Tracing | | |
| **Trace to RS Criteria** | RS traceability will be added in next release | | |
| **Reference to Test Environment** | STC_LT_00001 | | |
| **Configuration Parameters** | - LT modules in both ECUs are configured properly.<br><br>- ECU ID for [ECU1] is set to ECU1<br><br>- [APP1] has LT Application ID APPID1.<br><br>- Context ID for [APP1] is set to CTX1<br><br>- ECU ID for [ECU2] is set to ECU2<br><br>- [APP2] has LT Application ID APPID2.<br><br>- 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.<br><br>- [APP1] default log level is set to Verbose.<br><br>- [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]<br><br>Receive log messages. | | Tester receives log messages every 0.5 seconds.<br><br>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]<br><br>Second LT Client connects to [ECU2] on Port 3490 using TCP. | | Client connected. |

▽

△

| | | |
|---|---|---|
| **Step 3** | [LT Tester]<br><br>Receive log messages | Messages from [ECU1] are still received every 0.5 seconds.<br><br>Tester additionally receives log messages from ECU2 every 0.5 seconds.<br><br>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 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.

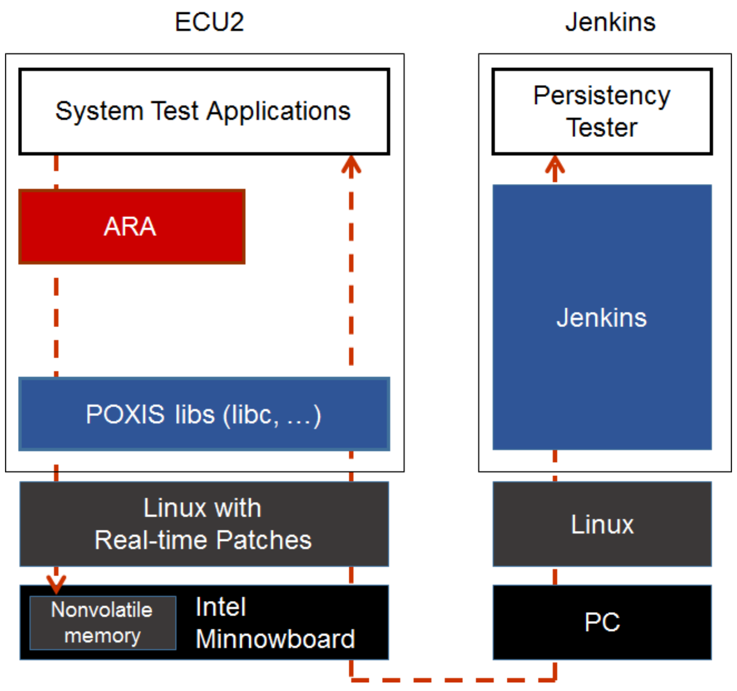

**Figure 8.1: Illustration of test setup for Persistency.**

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **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** | 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.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Store integer <intData> with associated key <intKey> in key-value database. | | |
| **Step 2** | [PTA]<br>Retrieve integer from key-value database using the associated key and store it in variable <retIntData>. | Originally written integer value is returned.<br>And values of <intData> and <retInt Data> 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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **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** | - Persistency tester is connected to ECU1.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Store float <floatData> with associated key <floatKey> in key-value database. | | |
| **Step 2** | [PTA]<br>Retrieve float from key-value database using the associated key and store it in variable <retFloatData>. | Originally written float value is returned.<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **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** | - Persistency tester is connected to ECU1.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Store string <stringData> with associated key <stringKey> in key-value database. | | |
| **Step 2** | [PTA]<br>Retrieve string from key-value database using the associated key and store it in variable <retStringData>. | | Originally written string value is returned.<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **Trace to RS Criteria** | [RS_PER_00004], [RS_PER_00010] | | |
| **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.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Write string <stringData> to file via file stream. | | |
| **Step 2** | [PTA]<br>Close file. | | |
| **Step 3** | [PTA]<br>Open file. | | File opened successfully. |
| **Step 4** | [PTA]<br>Retrieve string from file via file stream and store it in variable <retStringData>. | | Originally written string value is retrieved.<br>And Values of <stringData> and <ret StringData> 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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **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** | - Persistency tester is connected to ECU1.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Store integer <intData> with associated key <intKey> in key-value database. | |
| **Step 2** | [Persistency Tester]<br>Request reboot. | |
| **Step 3** | [Persistency Tester]<br>Wait until ECU1 has rebooted and PTA is initialized. | |
| **Step 4** | [PTA]<br>Open database. | Database file is opened. |
| **Step 5** | [PTA]<br>Retrieve integer from key-value database using the associated key and store it in variable <retIntData>. | Originally written integer value is returned.<br>And Values of <intData> and <retInt Data> 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 | **State** | Draft |
| **Affected Functional Cluster** | Persistency | | |
| **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** | - Persistency tester is connected to ECU1.<br>- Software components on ECU1 are initialized.<br>- 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]<br>Write string <stringData> to file via file stream. | |
| **Step 2** | [PTA]<br>Close file. | |
| **Step 3** | [Persistency Tester]<br>Request reboot. | |
| **Step 4** | [Persistency Tester]<br>Wait until ECU1 has rebooted and PTA is initialized. | |
| **Step 5** | [PTA]<br>Open file. | File opened successfully. |
| **Step 6** | [PTA]<br>Retrieve string from file via file stream and store it in variable <retStringData>. | Originally written string value is retrieved.<br>And Values of <stringData> and <ret StringData> 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

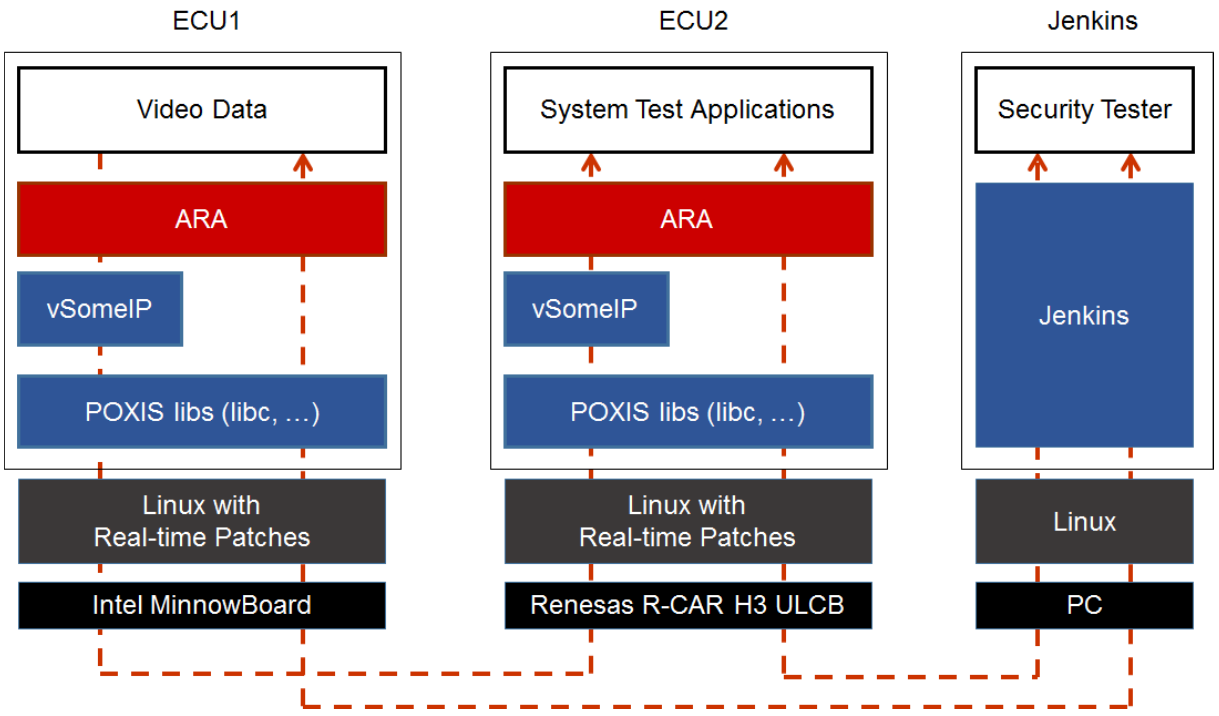

**Figure 9.1: Illustration of test setup for Security.**

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Security | | |
| **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** | - [APP1] offers and registers [SERVICE1], [SERVICE2], and [SERVICE3]<br>- [APP2] is authorized to use [SERVICE2] but not [SERVICE1] and [SERVICE3]<br>- [APP3] is authorized to use [SERVICE3] but not [SERVICE1] and [SERVICE2] | | |
| **Summary** | - [APP2] can successfully use [SERVICE2] but fails to use [SERVICE1] and [SERVICE3]<br>- [APP3] can successfully use [SERVICE3] but fails to use [SERVICE1] and [SERVICE2] | | |

▽

△

| | | |
|---|---|---|
| **Pre-conditions** | - Security Tester is connected to [ECU1]<br>- Software components on [ECU1] are initialized.<br>- [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]<br>Offers service [SERVICE1] | |
| **Step 2** | [APP1]<br>Offers service [SERVICE2] | |
| **Step 3** | [APP1]<br>Offers service [SERVICE3] | |
| **Step 4** | [APP2]<br>Requests service [SERVICE2] | Service discovery callback with a handle for [SERVICE2] is received by [APP2]. |
| **Step 5** | [APP3]<br>Requests service [SERVICE3] | Service discovery callback with a handle for [SERVICE3] is received by [APP3]. |
| **Step 6** | [APP2]<br>Requests service [SERVICE1] | Service is not available. |
| **Step 7** | [APP2]<br>Requests service [SERVICE3] | Service is not available. |
| **Step 8** | [APP3]<br>Requests service [SERVICE1] | Service is not available. |
| **Step 9** | [APP3]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Security | | |
| **Trace to RS Criteria** | [RS_IAM_00002], [RS_IAM_00007], [RS_IAM_00012] | | |
| **Reference to Test Environment** | STC_SEC_00001 | | |
| **Configuration Parameters** | - [APP1] offers and registers [SERVICE1] and is authorized to send [EVENT11] and [EVENT12]<br>- [APP2] offers and registers [SERVICE2] and is authorized to send [EVENT21] but not [EVENT22]<br>- [APP3] is authorized to subscribe for [EVENT11] and [EVENT21] | | |

▽

△

| | | |
|---|---|---|
| **Summary** | - [APP1] can successfully send [EVENT11] and [EVENT12]<br>- [APP2] can successfully send [EVENT21] but fails to send [EVENT22]<br>- [APP3] can successfully receive [EVENT11] from [APP1] and [EVENT21] from [APP2]<br>- [APP3] fails to receive [EVENT12] from [APP1] and [EVENT22] from [APP2] | |
| **Pre-conditions** | - Security Tester is connected to [ECU1]<br>- Software components on [ECU1] are initialized.<br>- [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]<br>Offers service [SERVICE1] with [EVENT11] and [EVENT12] | |
| **Step 2** | [APP2]<br>Offers service [SERVICE2] with [EVENT21] | |
| **Step 3** | [APP3]<br>Subscribes for [EVENT11] | Subscription is successful. |
| **Step 4** | [APP3]<br>Subscribes for [EVENT21] | Subscription is successful. |
| **Step 5** | [APP1]<br>Sends [EVENT11] | [APP3] receives notification for [EVENT11] |
| **Step 6** | [APP2]<br>Sends [EVENT22] | Event is dropped silently. [APP2] is not notified. |
| **Step 7** | [APP2]<br>Sends [EVENT21] | [APP3] receives notification for [EVENT21] |
| **Step 8** | [APP1]<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Security | | |
| **Trace to RS Criteria** | [RS_IAM_00002], [RS_IAM_00007], [RS_IAM_00012] | | |
| **Reference to Test Environment** | STC_SEC_00001 | | |

▽

△

| | | |
|---|---|---|
| **Configuration Parameters** | - [APP1] offers and registers [SERVICE1] and is authorized to send [EVENT11] and [EVENT12]<br>- [APP2] offers and registers [SERVICE2] and is authorized to send [EVENT21] but not [EVENT22]<br>- [APP3] is authorized to receive [EVENT11] | |
| **Summary** | - [APP1] can successfully send [EVENT11] and [EVENT12]<br>- [APP2] can successfully send [EVENT21] but fails to send [EVENT22]<br>- [APP3] can successfully receive [EVENT11] from [APP1]<br>- [APP3] fails to subscribe for [EVENT12], [EVENT21] and [EVENT22] | |
| **Pre-conditions** | - Security Tester is connected to [ECU1]<br>- Software components on [ECU1] are initialized.<br>- [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]<br>Offers service [SERVICE1] with [EVENT11] and [EVENT12] | |
| **Step 2** | [APP2]<br>Offers service [SERVICE2] with [EVENT21] | |
| **Step 3** | [APP3]<br>Subscribes for [EVENT11] | Subscription is successful. |
| **Step 4** | [APP1]<br>Sends [EVENT11] | [APP3] receives notification for [EVENT11] |
| **Step 5** | [APP1]<br>Sends [EVENT12] | [EVENT12] is dropped and [APP3] does not receive notification for [EVENT12] |
| **Step 6** | [APP2]<br>Sends [EVENT21] | [EVENT21] is dropped and [APP3] does not receive notification for [EVENT21] |
| **Step 7** | [APP2]<br>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

The 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. The System Test Specification checks 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 is running the job with the UCM Tester which is connected via Ethernet to the [ECU1] which is hosting the UCM Test Application (UTA).

The UCM Tester is supposed to check the pass criteria.

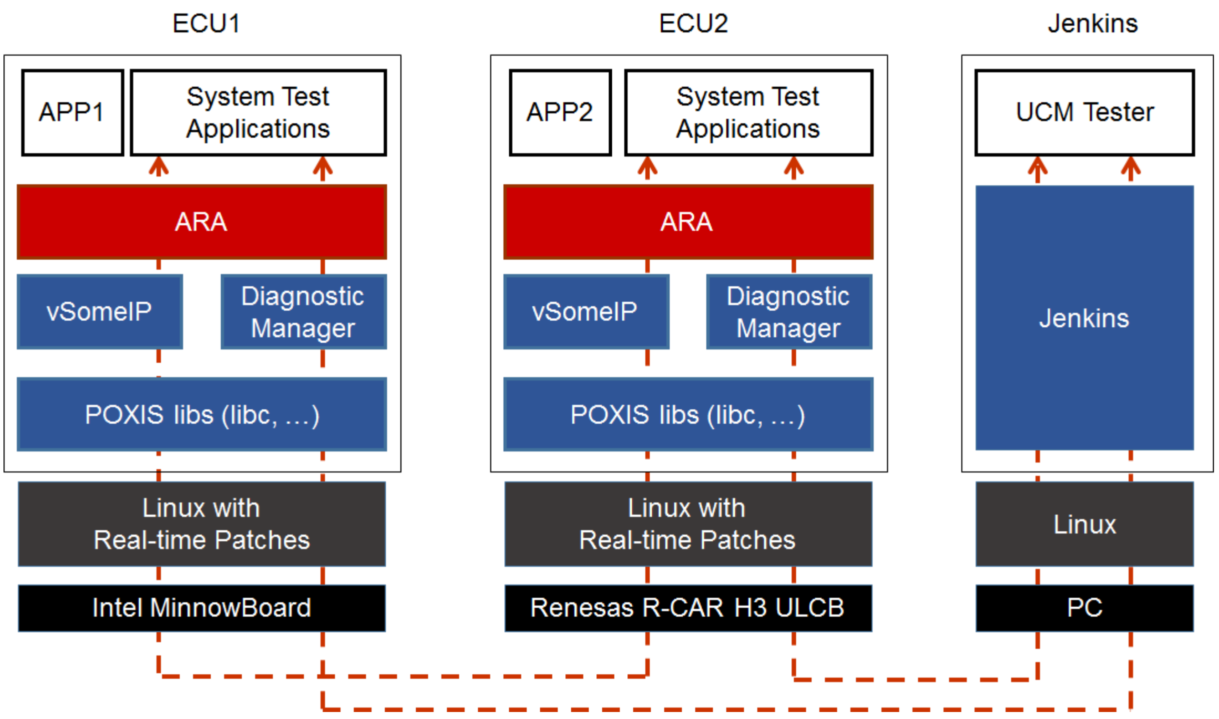

**Figure 10.1: Illustration of test setup for Update and Configuration Management.**

## 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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Trace to RS Criteria** | [RS_UCM_00010], [RS_UCM_00002] | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [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 updates are available. If any updates are available, present the list of available SW packages to the user. User then selects the required package and requests UTA to download the requested package. | | |
| **Pre-conditions** | - UCM Tester is connected to [ECU1].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send a request to [UTA] to read current SW version and name from UCM. | | |
| **Step 2** | [UTA]:<br>Start the mechanism to query read current SW version / name from UCM. | | |
| **Step 3** | [UCMTester]:<br>Receive response from [UTA] and store it in <UCM_SWVersion>. | | Payload of response contains SW version and name from UCM. |
| **Step 4** | [UCMTester]:<br>Send a request to [UTA] to read available SW version and name from backend system. | | |
| **Step 5** | [UTA]:<br>Start mechanism to read all available SW version/name list. | | |
| **Step 6** | [UCMTester]:<br>Receive response from [UTA] and store it in <backend_SWVersion_List>. | | |
| **Step 7** | [UCMTester]:<br>Send a request to download package <xyz> from available SW version/name list received from backend system. | | |
| **Step 8** | [UTA]:<br>Start mechanism to download SW package as per specified in the request. | | Requested package is downloaded successfully. |

▽

△

| | | |
|---|---|---|
| **Step 9** | [UCMTester]:<br>Send a request to read list of downloaded SW packages. | |
| **Step 10** | [UTA]:<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Trace to RS Criteria** | [RS_UCM_00011], [RS_UCM_00003], [RS_UCM_00023], [RS_UCM_00017], [RS_UCM_00030] | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [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].<br>- Software components on [ECU1] are initialized.<br>- [ECU1] is in Machine State Parking.<br>- SW package is downloaded and available locally to be updated. | | |
| **Post-conditions** | - TCP connection between UCM Tester and [ECU1] is closed. | | |
| **Main Test Execution** | | | |
| **Test Steps** | | **Pass Criteria** | |
| **Step 1** | [UCMTester]:<br>Send request to check availability of resources for data transfer. | | |
| **Step 2** | [UTA]:<br>Start mechanism to check availability of resources. | If result == success. | |
| **Step 3** | [UCMTester]:<br>Send request(trigger from user) to update a SW package. | | |
| **Step 4** | [UTA]:<br>Starts mechanism to initialize it for approval. | Send an ACK message after successful initialization for performing an update. | |
| **Step 5** | [UCMTester]:<br>Send request (user approval) to update a SW package as per package manifest (SW version and name). | | |
| **Step 6** | [UTA]:<br>Start mechanism to update a SW package. | | |

▽

△

| | | |
|---|---|---|
| **Step 7** | [UCMTester]:<br>Send a request to read progress status of an update. | ACK from UCM after successful update of SW package. |
| **Step 8** | [UTA]:<br>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]:<br>Receive response of succssful update of the package. | |
| **Step 10** | [UCMTester]:<br>Send request to activate updated SW package. | |
| **Step 11** | [UTA]:<br>Start mechanism to check SW package dependencies. | |
| **Step 12** | [UCMTester]:<br>Receive response of successful activation. | |
| **Step 13** | [UTA]:<br>Read value of persistent data associated with the SW package. | Persistent data is updated in kvs database by UCM as expected. |
| **Step 14** | [UCMTester]:<br>Send request (user approval)to update a SW package as per package manifest (SW version and name). | |
| **Step 15** | [UTA]:<br>Start mechanism to update a SW package. | |
| **Step 16** | [UCMTester]:<br>Send request to read progress status of an update. | |
| **Step 17** | [UCMTester]:<br>Start mechanism to provide progress status of an update of the SW package. | |
| **Step 18** | [UCMTester]:<br>Receive response of unsuccessful update of the SW package. | |
| **Step 19** | [UCMTester]:<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **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** | - [UTA] is configured.<br>- [Diagnostic module] is configured. | |
| **Summary** | UTA has the SW package available which is to be installed. UCM Tester 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].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to check availability of resources for data transfer. | |
| **Step 2** | [UTA]:<br>Start mechanism to check availability of resources and return result based on availability of resource. | Result == success. |
| **Step 3** | [UCMTester]:<br>Send request (user approval) to install a SW package as per package manifest (SW version/name). | |
| **Step 4** | [UTA]:<br>Start mechanism to install a SW package and write/store persistent data associated with the SW package. | |
| **Step 5** | [UCMTester]:<br>Response of successful installation of package. | ACK from UCM after successful installation of SW package. |
| **Step 6** | [UCMTester]:<br>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]:<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Trace to RS Criteria** | [RS_UCM_00004], [RS_UCM_00005], [RS_UCM_00018] | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |

▽

△

| | | |
|---|---|---|
| **Configuration Parameters** | - [UTA] is configured.<br>- [Diagnostic module] is configured. | |
| **Summary** | UTA has the information about the SW package to be uninstalled. UCM Tester 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].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>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]:<br>Start mechanism to uninstall a SW package. | |
| **Step 3** | [UCMTester]:<br>Response of successful uninstallation of package. | ACK from UCM after successful uninstallation of SW package. |
| **Step 4** | [UCMTester]:<br>Send request (trigger from user) to uninstall a SW package as per package manifest. | |
| **Step 5** | [UTA]:<br>Start mechanism to uninstall a SW package. | |
| **Step 6** | [UCMTester]:<br>Response of unsuccessful installation of package. | NACK from UCM after unsuccessful installation of SW package. |
| **Step 7** | [UTA]:<br>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. Verification that ,Rollback to arbitrary SW shall be prevented. | | |
| **ID** | STS_UCM_00005 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Trace to RS Criteria** | [RS_UCM_00008], [RS_UCM_00001], [RS_UCM_00023], [RS_UCM_00031] | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |

▽

△

| | | |
|---|---|---|
| **Configuration Parameters** | - [UTA] is configured.<br>- [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].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to install a SW package as per package manifest. | |
| **Step 2** | [UTA]:<br>Start mechanism to install a SW package. | |
| **Step 3** | [UCMTester]:<br>Send request to get SW package installation status. | |
| **Step 4** | [UTA]:<br>Start mechanism to get installation status of a requested SW package. | |
| **Step 5** | [UCMTester]:<br>Receive response of installation status. | Installation status is received as Failed. |
| **Step 6** | [UCMTester]:<br>Send request to perform Rollback to previous - x SW version. | |
| **Step 7** | [UTA]:<br>Start mechanism to Rollback to previous - x SW version. | |
| **Step 8** | [UCMTester]:<br>Receive response of unsuccessful Rollback. | NACK for unsuccessful Rollback. |
| **Step 9** | [UCMTester]:<br>Send request to Rollback to previous SW package version. | |
| **Step 10** | [UTA]:<br>Start mechanism to Rollback to latest available SW package. | |
| **Step 11** | [UCMTester]:<br>Receive response of successful Rollback. | 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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Trace to RS Criteria** | [RS_UCM_00032] | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [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** | - UCM Tester is connected to [ECU1].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to read update history of an adaptive platform. | | |
| **Step 2** | [UTA]:<br>Start mechanism to read Update history of the platform. | ACK from UCM | |
| **Step 3** | [UCMTester]:<br>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]:<br>Send request to read update history of an adaptive platform. | | |
| **Step 5** | [UTA]:<br>Start mechanism to read update history of the platform. | NACK from UCM. | |
| **Step 6** | [UCMTester]:<br>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 mutiple clients can perform data transfer of SW packages, simultaneously. | | |
| **ID** | STS_UCM_00007 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Trace to RS Criteria** | [RS_UCM_00019] | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [UTA1] is configured.<br>- [Diagnostic module] is configured. | | |
| **Summary** | - UTA starts data transfer of SW package 1.<br>- UTA1 also starts data trasfer of SW package 2, simultaneously.<br>- UCM allows UTA /UTA1 to perform data trasnfer, simultaneously. | | |
| **Pre-conditions** | - UCM Tester is connected to [ECU1].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to UTA to transfer SW package 1. | | |
| **Step 2** | [UTA]:<br>Start mechanism to prepare for accepting SW package 1. | | |
| **Step 3** | [UCMTester]:<br>Send request to UTA1 for data transfer of SW package 2. | | |
| **Step 4** | [UTA1]:<br>Start mechanism to prepare for accepting SW package 2. | | |
| **Step 5** | [UCMTester]:<br>Send a request to get information about transferred SW package list. | | |
| **Step 6** | [UTA/UTA1]:<br>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 mutiple clients can perform Install/Update/Removal of SW packages, sequentially. | | |
| **ID** | STS_UCM_00008 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Trace to RS Criteria** | [RS_UCM_00024], [RS_UCM_00026], [RS_UCM_00002] | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [UTA1] is configured.<br>- [Diagnostic module] is configured. | | |
| **Summary** | - UTA queries UCM to install/update/remove SW package 1, UTA1 also queries UCM to install/update/remove SW package 2 ,simultaneously.<br>- UCM rejects install/update/removal request from UTA1. UTA1 has to wait untill UTA finishes install/update/removal of SW package 1. | | |
| **Pre-conditions** | - UCM Tester is connected to [ECU1].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to read current SW version. | | |
| **Step 2** | [UTA]:<br>Start mechanism to provide current SW version. | | |
| **Step 3** | [UCMTester]:<br>Receive response of current SW version and store it in <var1>. | | |
| **Step 4** | [UCMTester]:<br>Send a request to install/update/remove SW Package 1 to UTA. | | |
| **Step 5** | [UTA]:<br>Start mechanism to install/update/remove SW package 1. | | |
| **Step 6** | [UCMTester]:<br>Send a request to read current SW version to UTA1. | | |
| **Step 7** | [UTA1]:<br>Start mechanism to provide current SW version. | | |
| **Step 8** | [UCMTester]:<br>Receive response as a SW version and store it in <var2>. | | |
| **Step 9** | [UCMTester]:<br>Send a request to install/update/remove SW package 2 to UTA1. | | |
| **Step 10** | [UTA1]:<br>Start mechanism to install/update/remove SW package. | | |

▽

△

| | | |
|---|---|---|
| **Step 11** | [UCMTester]:<br>Receive response as status of install/update/removal. | Status = Reject. |
| **Step 12** | [UCMTester]:<br>Send a request to UTA1 to get current status of UCM. | |
| **Step 13** | [UTA1]:<br>Start mechanism to provide UCM state. | |
| **Step 14** | [UCMTester]:<br>Receive response as UCM state .If state is Busy ,wait untill state changes to READY. | UCMState = Busy/READY. |
| **Step 15** | [UCMTester]:<br>Send request to UTA1 to install/update/removal SW package 2. | |
| **Step 16** | [UTA1]:<br>Start mechanism to prepare for install/update/removal of SW package 2. | |
| **Step 17** | [UCMTester]:<br>Receive response as successful install/update/removal of SW package 2. | |
| **Step 18** | [UCMTester]:<br>Send a request to read SW version. | |
| **Step 19** | [UTA1]:<br>Start mechanism to send SW version of newly installed SW package. | |
| **Step 20** | [UCMTester]:<br>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 | **State** | Draft |
| **Affected Functional Cluster** | Update and Configuration Management | | |
| **Reference to Test Environment** | STC_UCM_00001 | | |
| **Trace to RS Criteria** | [RS_UCM_00020], [RS_UCM_00002], [RS_UCM_00003] | | |
| **Configuration Parameters** | - [UTA] is configured.<br>- [Diagnostic module] is configured. | | |
| **Summary** | - UTA queries UCM to install/Update a SW package 2.<br>- UTA later realises that there are some discrepancies, it issues Cancel request to cancel ongoing Install/Update of SW package. | | |

▽

△

| | | |
|---|---|---|
| **Pre-conditions** | - UCM Tester is connected to [ECU1].<br>- Software components on [ECU1] are initialized.<br>- [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]:<br>Send request to read current version of the installed SW package. | |
| **Step 2** | [UTA]:<br>Start mechanism to provide current version of SW package. | |
| **Step 3** | [UCMTester]:<br>Receive response of current SW version and store it in <var1>. | |
| **Step 4** | [UCMTester]:<br>Send a request to install/update SW package 2. | |
| **Step 5** | [UTA]:<br>Start mechanism to install/update SW Package 2. | |
| **Step 6** | [UCMTester]:<br>Send a request to cancel ongoing install/update of SW package 2. | |
| **Step 7** | [UTA]:<br>Prepare to cancel ongoing operation and send an ACK for successful cancellation. | |
| **Step 8** | [UCMTester]:<br>Send a request to read SW version. | |
| **Step 9** | [UTA]:<br>Start mechanism to provide SW version. | |
| **Step 10** | [UCMTester]:<br>Receive response of current SW version. | <var1> and <var2> are equal (New SW package 2 install/update is cancelled succesfully). |

# 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 |

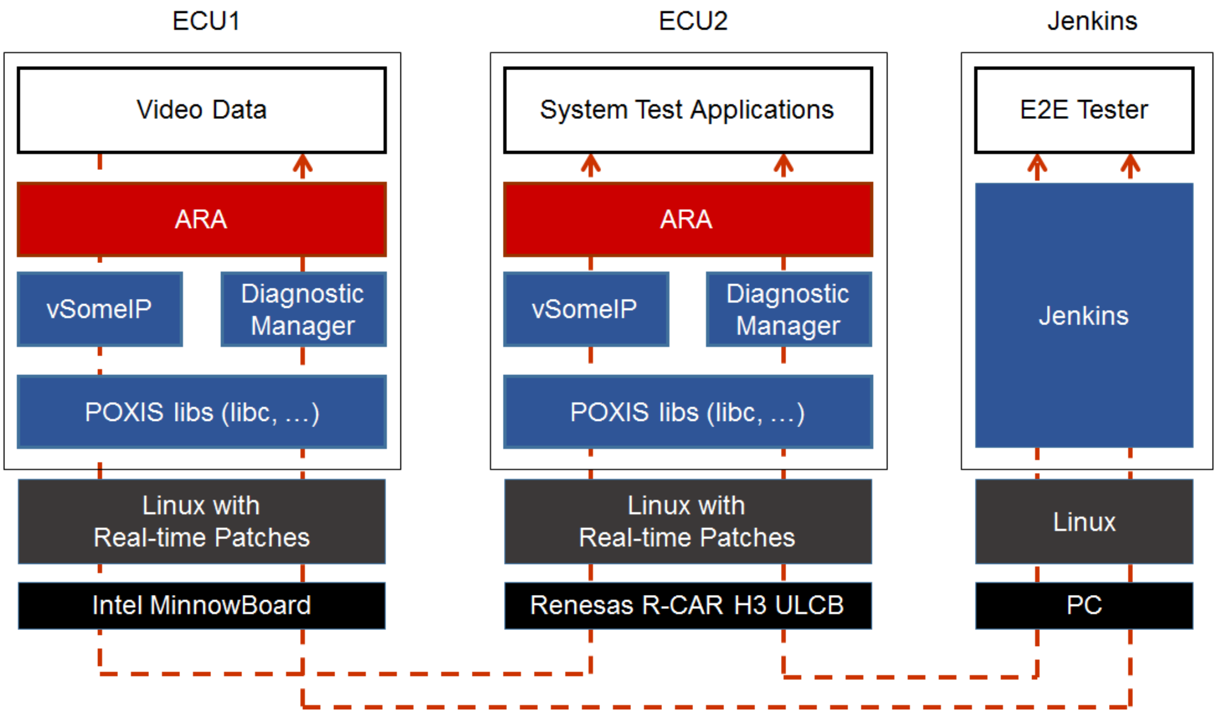

**Figure 11.1: Illustration of test setup for STC-E2E-00001.**

| | |
|---|---|
| **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 |

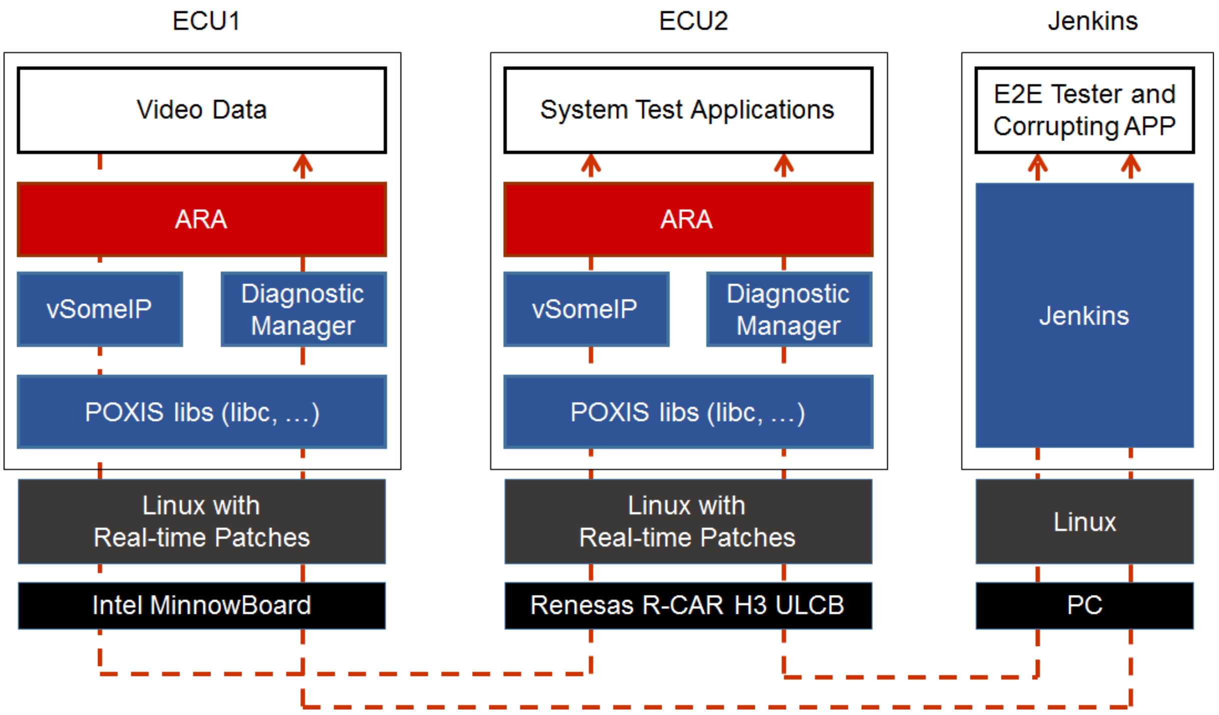

**Figure 11.2: Illustration of test setup for STC-E2E-00002.**

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 | **State** | Draft |
| **Affected Functional Cluster** | Safety | | |
| **Trace to RS Criteria** | [RS_E2E_08539], [RS_E2E_08541], [RS_E2E_08543] | | |
| **Reference to Test Environment** | STC_E2E_00001 | | |
| **Configuration Parameters** | - Event based communication.<br>- The existing communication services comprise the following (service & data names are arbitrary):<br>- [SERVICE1]: Offered by [APP1], requested by [APP2].<br>- <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** | - [E2E Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [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** | [E2E Tester]<br>Request for change of Machine State to Driving from Execution Manager.<br>Machine State for ECU1 and ECU2 are changed to Driving, and [APP 1] and [APP 2] are started up. | |
| **Step 2** | [APP1]<br>Offer service [SERVICE1]. | |
| **Step 3** | [APP2]<br>Request service [SERVICE1]. | |
| **Step 4** | [APP1]<br>Send E2E protected <Data1> with arbitrary values | |
| **Step 5** | [APP2]<br>Calls GetCheckStatus() for <Data1> | [APP2] reads CheckStatus = Ok |
| **Step 6** | [APP2]<br>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<br>Include 4 kbyte length data for the above. | CheckStatus is always = Ok<br><Data1> is always received with correct values |

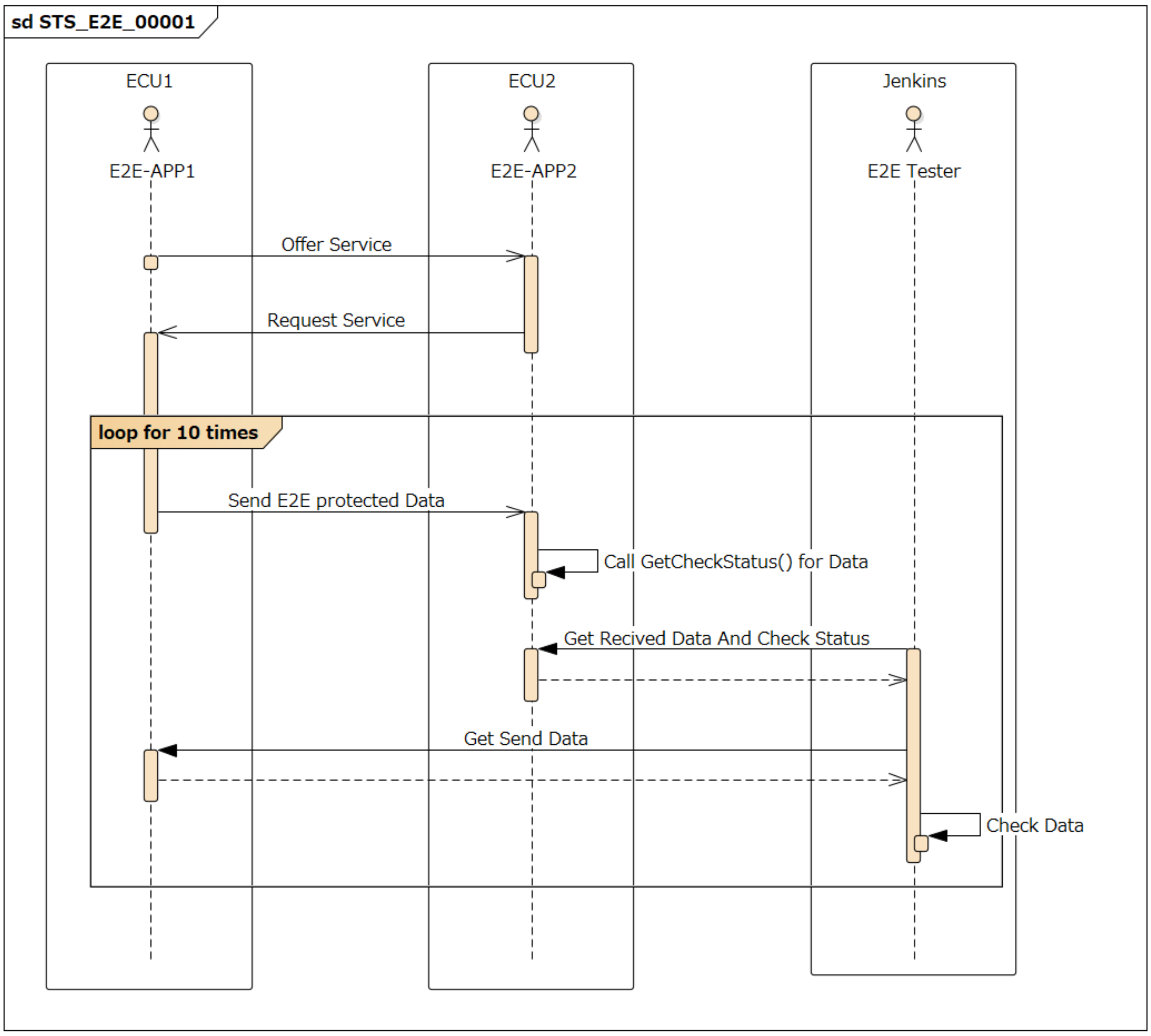

**Figure 11.3: Sequence diagram of STC-E2E-00001.**

### 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 | **State** | Draft |
| **Affected Functional Cluster** | Safety | | |
| **Trace to RS Criteria** | [RS_E2E_08534] | | |
| **Reference to Test Environment** | STC_E2E_00002 | | |

▽

△

| | | |
|---|---|---|
| **Configuration Parameters** | - Event based communication.<br>- The existing communication services comprise the following (service & data names are arbitrary):<br>- [SERVICE1]: Offered by [APP1], requested by [APP2].<br>- <Data1> is E2E protected, sent by [APP1], received by [APP2].<br>- [Corrupting APP] to send <Data1>, with similar message format as sent by [APP1] | |
| **Summary** | [SERVICE1] & <Data1> are offered/sent by [APP1] on ECU1, and they are used/received by [APP2] on ECU2.<br>When [Corrupting APP] sends the same communication sent by [APP1], but with some corrupted data, other apps detect this thanks to the E2E protection. | |
| **Pre-conditions** | - [E2E Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [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** | [E2E Tester]<br>Request for change of Machine State to Driving from Execution Manager.<br>Machine State for ECU1 and ECU2 are changed to Driving, and [APP 1] and [APP 2] are started up. | |
| **Step 2** | [APP1]<br>Offer service [SERVICE1]. | |
| **Step 3** | [APP2]<br>Request service [SERVICE1]. | |
| **Step 4** | [APP1]<br>Send E2E protected <Data1> with arbitrary values | |
| **Step 5** | [APP2]<br>Executes:<br>• GetCheckStatus() for <Data1><br>Update for <Data1> | [APP2]<br>• reads CheckStatus = Ok<br>• receives correct value of <Data1> |
| **Step 6** | [CorruptingApp]<br>Sends the same Ethernet frame that was sent by [APP1], but with different value of <Data1> | [APP2] is notified of CRC error while receiving <Data1> (CheckStatus = Error) |
| **Step 7** | [CorruptingApp]<br>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) | [APP2] is notified of Counter error while receiving <Data1> (CheckStatus = WrongSequence) |

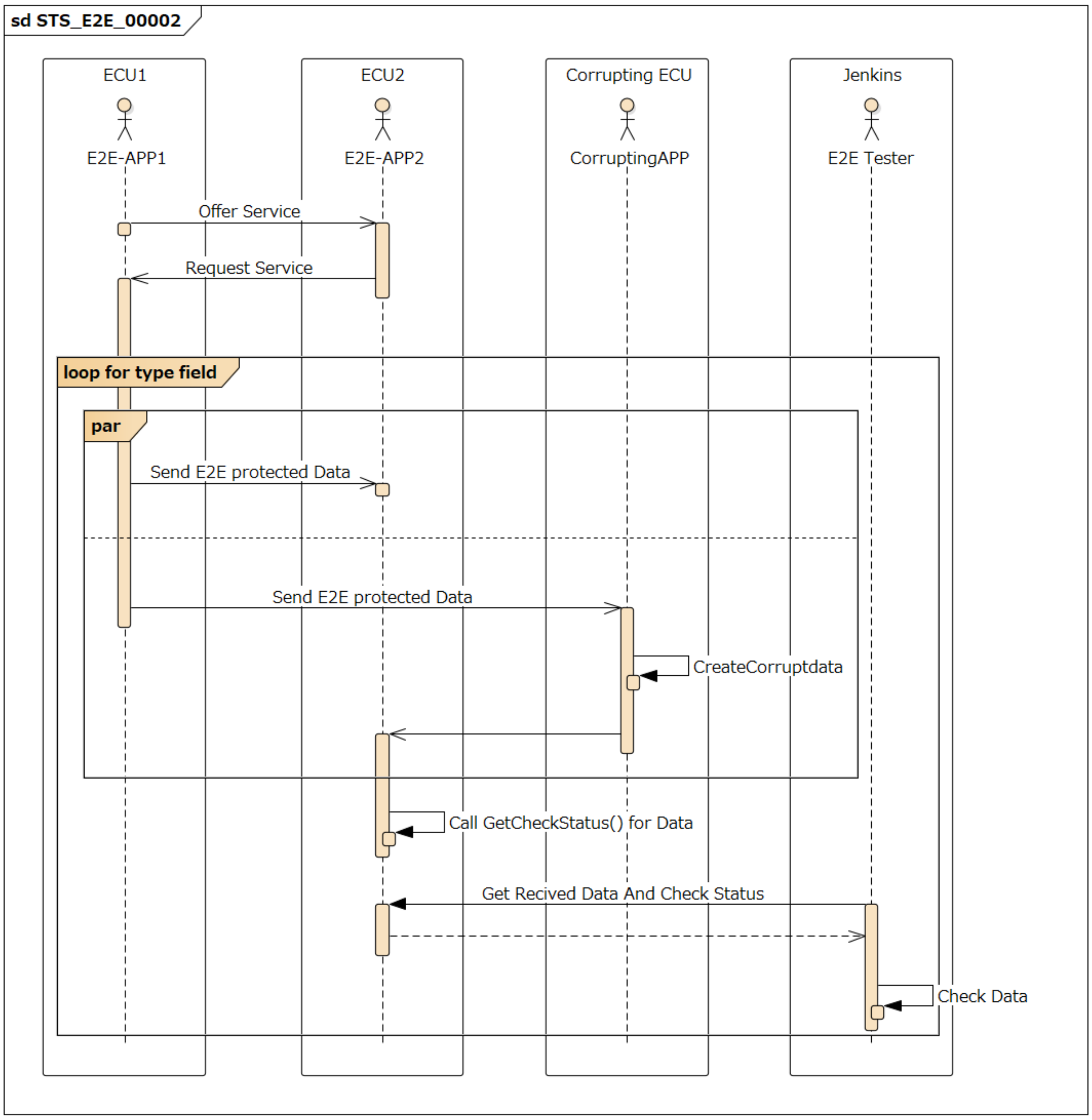

**Figure 11.4: Sequence diagram of STC-E2E-00002.**

# 12 Test configuration and test steps for Time Synchronization

## 12.1 Test System

### 12.1.1 Test configurations

| | |
|---|---|
| **Configuration ID** | STC_TS_00001 |
| **Description** | Standard Jenkins server for Time Synchronization 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 |

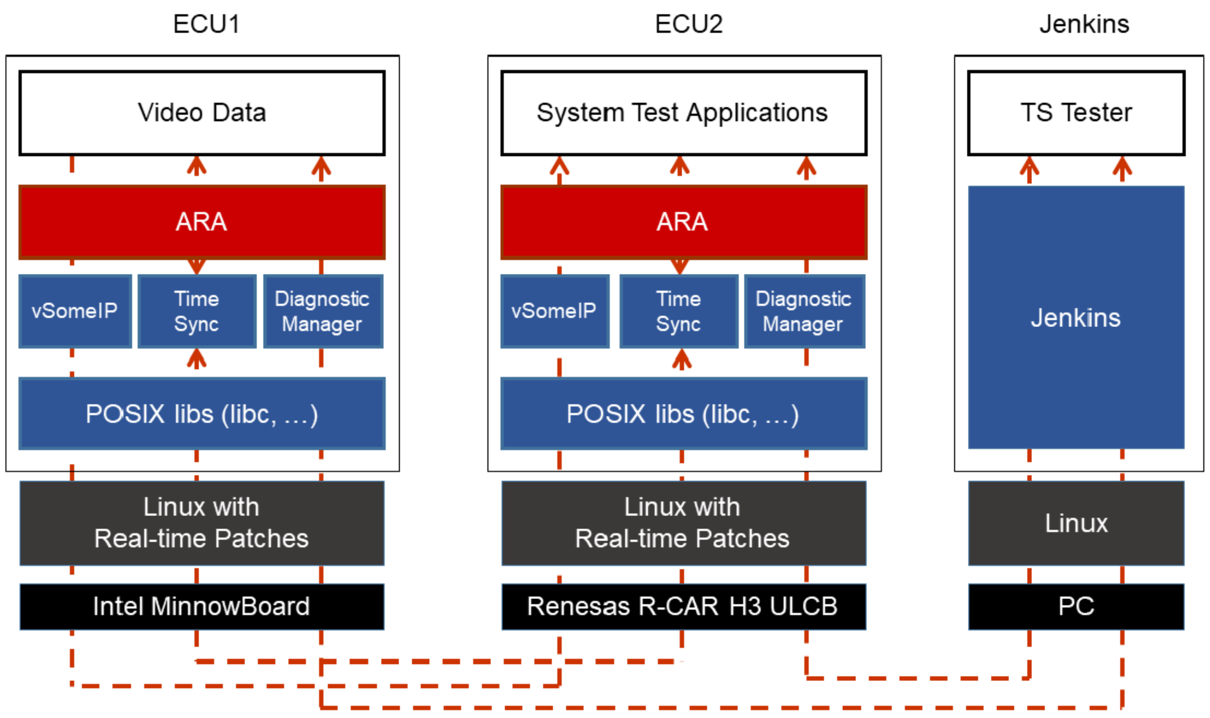

**Figure 12.1: Illustration of test setup for Time Synchronization.**

The Jenkins Server, running the job with the Time Synchronization test ([TS Tester]) is connected via Ethernet to [ECU1] hosting the System Test Application [APP1] and [ECU2] hosting the System Test Application [APP2].

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

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

## 12.2 Test cases

### 12.2.1 [STS_TS_00001] Checking APIs of a Offset Slave TimeBase (TB)

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that whether APIs of a Offset Slave TB can be used correctly. | | |
| **ID** | STS_TS_00001 | **State** | Draft |
| **Affected Functional Cluster** | Time Synchronization | | |
| **Trace to RS Criteria** | [RS_TS_00001], [RS_TS_00005], [RS_TS_00012], [RS_TS_00013], [RS_TS_00017], [RS_TS_00021], [RS_TS_00026] | | |
| **Reference to Test Environment** | STC_TS_00001 | | |
| **Configuration Parameters** | - [ECU1] is synced by [ECU2].<br>- [ECU2] is Global Time Master.<br>- [ECU1] has a Offset Slave TB and a Synchronized Slave TB.<br>- [ECU2] has a Offset Master TB and a Synchronized Master TB.<br>- The Synchronized Slave TB on [ECU1] is synced by the Synchronized Master TB on [ECU2].<br>- The Offset Slave TB on [ECU1] depend on the Synchronized Slave TB on [ECU1],<br>- The Offset Master TB on [ECU2] depend on the Synchronized Mater TB on [ECU2]. | | |
| **Summary** | Verification that [APP1] can use APIs of Offset Slave TB. | | |
| **Pre-conditions** | - [TS Tester] is connected to [ECU1].<br>- [ECU1] is in Machine State Parking.<br>- [APP1] is shut down according to Machine State. | | |
| **Post-conditions** | [TS Tester] is disconnected to [ECU1]. | | |
| **Main Test Execution** | | | |
| **Test Steps** | | **Pass Criteria** | |
| **Step 1** | [TS Tester]<br>Request for change of Machine State to Driving from Execution Manager.<br>Machine State for [ECU1] is changed to Driving, and [APP 1] is started up. | | |
| **Step 2** | [APP1]<br>Find the Offset Slave TB on [ECU1]. | The Offset Slave TB on [ECU1] is found successfully. | |
| **Step 3** | [APP1]<br>Configure the Offset Slave TB on [ECU1]. | | |
| **Step 4** | [APP1]<br>Get rate deviation of the Offset Slave TB on [ECU1]. | Rate deviation is got successfully. | |
| **Step 5** | [APP1]<br>Get TimeBaseStatus of the Offset Slave TB on [ECU1]. | TimeBaseStatus is got successfully. | |
| **Step 6** | [APP1]<br>Get a getType of the Offset Slave TB on [ECU1]. | The getType is Offset Slave TB. | |
| **Step 7** | [APP1]<br>Set Offset value of the Offset Slave TB on [ECU1]. | | |

▽

△

| | | |
|---|---|---|
| **Step 8** | [APP1]<br>Get Offset value of the Offset Slave TB on [ECU1]. | Offset value is the value set in Step 7. |
| **Step 9** | [APP1]<br>Get current time of the Offset Slave TB on [ECU1]. | Current time is got successfully. |
| **Step 10** | [APP1]<br>Start the timer of the Offset Slave TB on [ECU1] so that the timer will expire at the specified time. | |
| **Step 11** | [APP1]<br>When time-up is notified. Get current time of the Offset Slave TB on [ECU1]. | Current time is the specified time. |

### 12.2.2 [STS_TS_00002] TimeSynchronization of applications between ECUs.

| | | | |
|---|---|---|---|
| **Test Objective** | Verification that synchronization between the application on [ECU1] and [ECU2] can correctly be done. | | |
| **ID** | STS_TS_00002 | **State** | Draft |
| **Affected Functional Cluster** | Time Synchronization | | |
| **Trace to RS Criteria** | [RS_TS_00005], [RS_TS_00020], [RS_TS_00026] | | |
| **Reference to Test Environment** | STC_TS_00001 | | |
| **Configuration Parameters** | - [ECU1] is synced by [ECU2].<br>- [ECU2] is Global Time Master.<br>- [ECU1] has a Offset Slave TimeBase(TB) and a Synchronized Slave TB.<br>- [ECU2] has a Offset Master TB and a Synchronized Master TB.<br>- The Synchronized Slave TB on [ECU1] is synced by the Synchronized Master TB on [ECU2].<br>- The Offset Slave TB on [ECU1] depend on the Synchronized Slave TB on [ECU1],<br>- The Offset Master TB on [ECU2] depend on the Synchronized Mater TB on [ECU2].<br>- Event based communication.<br>- The existing communication services comprise the following (service & data names are arbitrary):<br>• [SERVICE1]: Offered by [APP1], requested by [APP2].<br>• [SERVICE1]: [APP1] send a synchronization time to [APP2]. | | |
| **Summary** | Verification that [APP1] and [APP2] can be synchronized. | | |
| **Pre-conditions** | - [TS Tester] is connected to both ECUs.<br>- Both ECUs are in Machine State Parking.<br>- [APP1] and [APP2] are shut down according to Machine State. | | |
| **Post-conditions** | [TS Tester] is disconnected to both ECUs. | | |
| **Main Test Execution** | | | |
| **Test Steps** | | **Pass Criteria** | |

▽

△

| | | |
|---|---|---|
| **Step 1** | [TS Tester]<br>Request for change of Machine State to Driving from Execution Manager.<br>Machine State for [ECU1] and [ECU2] are changed to Driving, and [APP 1] and [APP 2] are started up. | |
| **Step 2** | [APP1]<br>Offer service [SERVICE1]. | |
| **Step 3** | [APP2]<br>Request service [SERVICE1]. | |
| **Step 4** | [APP1]<br>Find the Offset Slave TB on [ECU1]. | The Offset Slave TB on [ECU1] is found successfully. |
| **Step 5** | [APP1]<br>Configure the Offset Slave TB on [ECU1]. | |
| **Step 6** | [APP2]<br>Find the Offset Master TB on [ECU2]. | The Offset Master TB on [ECU2] is found successfully. |
| **Step 7** | [APP2]<br>Configure the Offset Master TB on [ECU2]. | |
| **Step 8** | [APP1]<br>Get current time of the Offset Slave TB on [ECU1]. | |
| **Step 9** | [APP1]<br>Decide a future synchronization time based on the current time so that [APP1] and [APP2] will be notified simultaneously and sync then. | |
| **Step 10** | [APP1]<br>Start the timer of the Offset Slave TB on [ECU1] so that the timer will expire at the synchronization time. | |
| **Step 11** | [APP1]<br>Send the synchronization time to [APP2]. | |
| **Step 12** | [APP2]<br>Receive the synchronization time from [APP1]. | |
| **Step 13** | [APP2]<br>Get current time of the Offset Master TB on [ECU2]. | |
| **Step 14** | [APP2]<br>Start the timer of the Offset Master TB on [ECU2] so that the timer will expire at the synchronization time. | |
| **Step 15** | [APP1][APP2]<br>Receive notify from the timer at the synchronization time. | |
| **Step 16** | [APP1][APP2]<br>Get the current time and store the current time. | Both current times are almost same. |

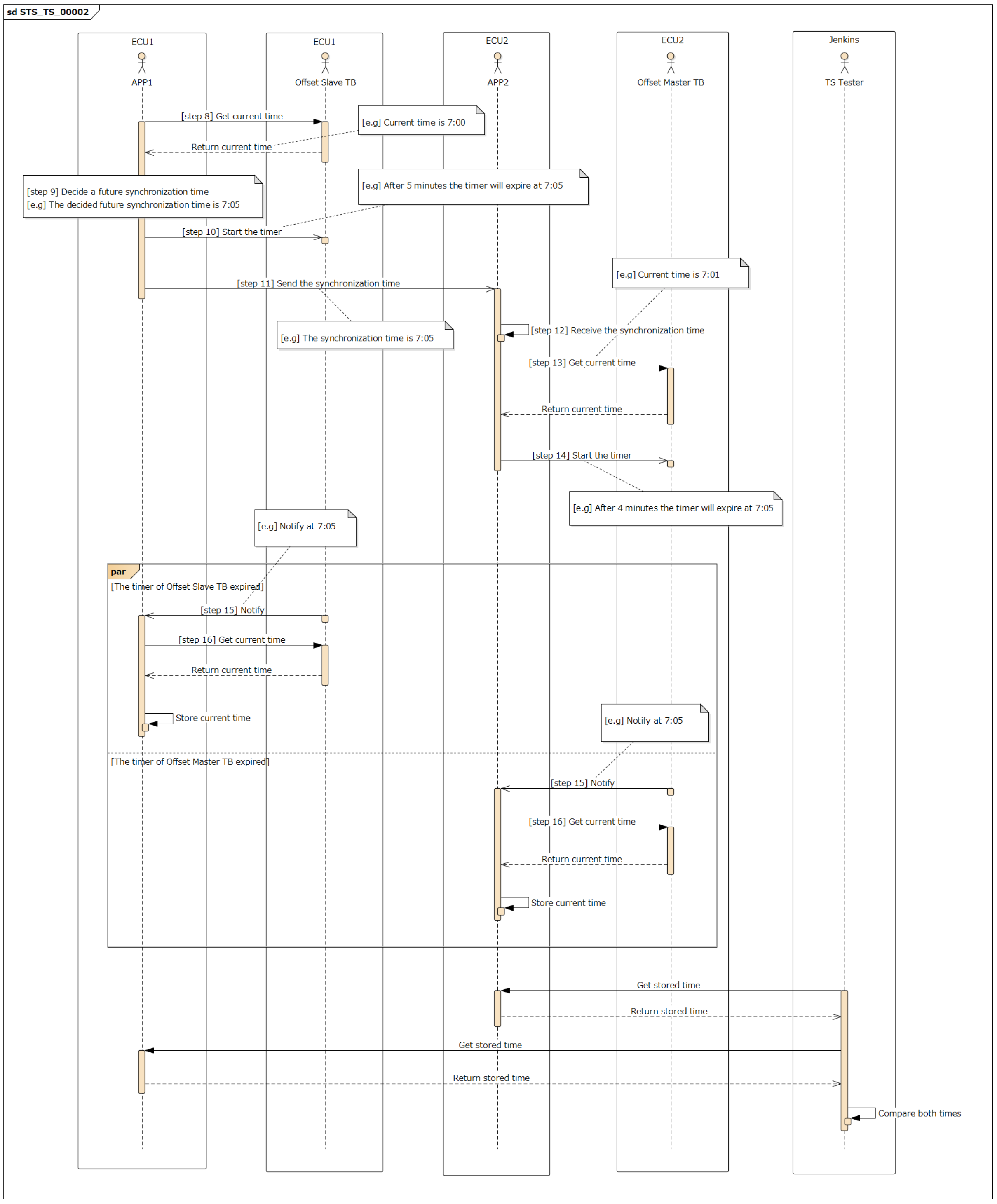
sd STS_TS_00002
ECU1
APP1
ECU1
Offset Slave TB
ECU2
APP2
ECU2
Offset Master TB
Jenkins
TS Tester
[step 8] Get current time
[e.g] Current time is 7:00
Return current time
[step 9] Decide a future synchronization time
[e.g] The decided future synchronization time is 7:05
[e.g] After 5 minutes the timer will expire at 7:05
[step 10] Start the timer
[e.g] Current time is 7:01
[step 11] Send the synchronization time
[e.g] The synchronization time is 7:05
[step 12] Receive the synchronization time
[step 13] Get current time
Return current time
[step 14] Start the timer
[e.g] After 4 minutes the timer will expire at 7:05
[e.g] Notify at 7:05
par
[The timer of Offset Slave TB expired]
[step 15] Notify
[step 16] Get current time
Return current time
Store current time
[e.g] Notify at 7:05
[The timer of Offset Master TB expired]
[step 15] Notify
[step 16] Get current time
Return current time
Store current time
Get stored time
Return stored time
Get stored time
Return stored time
Compare both times

**Figure 12.2: Sequence diagram of STC_TS_00002. [e.g] APP1 and APP2 sync at 7:05.**

# 13 References

[1] Glossary AUTOSAR_TR_Glossary