|  |  |
| --- | --- |
| **Open Issue** | **Issue description** |
| - | - |
|  |  |

Table of Contents

[1 DOCUMENT INTRODUCTION 3](#_Toc487725859)

[1.1 Purpose 3](#_Toc487725860)

[1.2 CDP2 Overall Connectivity architecture 3](#_Toc487725861)

[1.3 Scope 3](#_Toc487725862)

[1.4 References 3](#_Toc487725863)

[1.5 Terminology & Abbreviations 3](#_Toc487725864)

[2 Test Method 5](#_Toc487725865)

[3 Sampling Method 5](#_Toc487725866)

[4 Test Set-up 5](#_Toc487725867)

[5 Test Conditions 5](#_Toc487725868)

[6 Test Cases 6](#_Toc487725869)

[6.1 Manual Test Cases 6](#_Toc487725870)

[6.2 Automated Test Cases 20](#_Toc487725871)

[7 Test Instructions 23](#_Toc487725872)

[8 Revision History 23](#_Toc487725873)

[9 Approval 24](#_Toc487725874)

# DOCUMENT INTRODUCTION

## Purpose

This document describes the test cases (test scenarios) to be used for the verification of CommLib, demonstrating it meets the requirements, see ref. [SwRS], of CommLib.

## CDP2 Overall Connectivity architecture

Below picture shows the overall connectivity architecture and components. The components inside the blue box are described in this document.

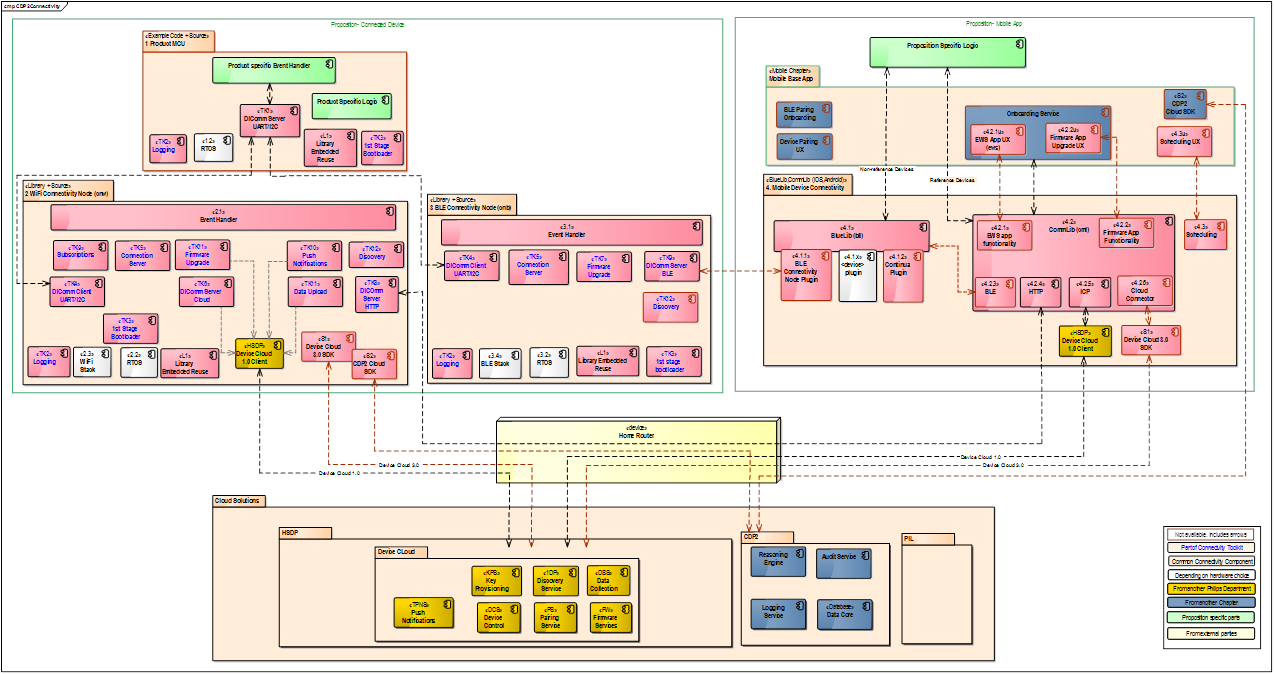


Figure 1 - CDP2 Overall Connectivity architecture

## Scope

This document applies to CommLib (Android and iOS), to be used in connected digital propositions.

## References

| **Reference** | **Identification** | **Title / additional remarks** |
| --- | --- | --- |
| [SwRS] | CML000001 | Requirements, CommLib |
| [DiComm] | JohSun-20160115-04V01 | SwIS DIComm Protocol |
| [SwTM] | CML000007 | RequirementsTraceability Matrix, CommLib |

## Terminology & Abbreviations

| **Terminology & Abbreviations** | **Description/Definition** |
| --- | --- |
| BLE | Bluetooth Low Energy |
| DiComm | Digital Interface Communication Protocol, see ref [DiComm] |
| Gherkin | Gherkin is a Business Readable, Domain Specific Language that lets you describe software’s behavior without detailing how that behavior is implemented. It is used to specify test scenarios that can be interpreted by tools like: cucumber and SpecFlow. Reference: The Cucumber Book (ISBN 978-1-93435-680-7) |
| TC | Test Case |

# Test Method

The test scenarios are written in the [Gherkin] design specification language. The [Gherkin] design specification language provides human-readable scenarios and steps (Given, When, Then) that could be executed either manually or automatically. The test scenarios shall verify that the requirements are met and shall be understandable by non-technical business owners.

A test scenario is part of a [Gherkin] feature file.  
A feature file contains one or more scenarios.

A requirement Id specified in the requirements document [SwRS] is tested on its key elements by one or more scenarios.

# Sampling Method

Not applicable. For the software tests no sampling method is applied.

# Test Set-up

Manual test cases require:

* Android and iOS phones
* Test Apps
* BLE Reference Node
* WiFI Reference Node
* Product Stub running on a Windows machine
* AirPurifier
* two WiFi networks with internet access.

The following Test Apps are used to verify CommLib:

* Commlib-ble-testapp (Android)
* Commlib-demoapp (Android)
* Commlib-explorer (Android)
* Commlib-demoapp (iOS)

Firmware revisions for Reference Nodes:

* BLE Reference Node firmware: 2.2.1
* WiFi Reference Node firmware: 2.2.0

BLE test scenarios must be executed with multiple Android phones from multiple manufacturers and a variation in Android releases.

# Test Conditions

The test conditions are fully contained inside the [Gherkin] test scenarios themselves.

For BLE test scenarios it is important that the physical distance between the phone and the BLE Reference Node is 50 cm. Changing the physical distance may impact the test results.

# Test Cases

The test cases are specified by using the Gherkin specification language. Each test case is a Gherkin Scenario.

The test cases (scenarios) used to verify the requirements of CommLib can be found in TFS under Test Plan: **Platform: cml\_bll Test design (Id:14718)**.

The requirements traceability matrix [SwTM] shows what requirement is verified by what test case (scenario).

## Manual Test Cases

|  |  |  |
| --- | --- | --- |
| Test Case ID | Test Case Title | Steps |
| 16043 | Non-discoverable peripheral is removed from list of discovered peripherals, when discovery is active | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 4. When peripheral becomes non-discoverable (e.g. out of reach) |
|  |  | Step 5. Then peripheral is removed from the list of discovered peripherals within 120 seconds |
| 16044 | Lost peripheral is rediscovered | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 4. And peripheral becomes non-discoverable (e.g. out of reach) |
|  |  | Step 5. And peripheral is removed from the list of discovered peripherals |
|  |  | Step 6. When peripheral becomes discoverable (e.g. within reach) |
|  |  | Step 7. Then peripheral is added to the list of discoverable peripherals within 120 seconds |
| 16098 | Discoverable peripheral is removed from list of discovered peripherals, when discovery is stopped | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 4. When application stops BLE discovery |
|  |  | Step 5. Then peripheral is removed from the list of discovered peripherals within 120 seconds |
| 16785 | Discoverable peripheral is added to list of discovered peripherals | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. When a peripheral becomes discoverable |
|  |  | Step 4. Then peripheral is added to the list of discovered peripherals within 120 seconds |
| 16790 | Discoverable peripheral stays in list of discovered peripherals | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 4. When peripheral remains discoverable for 2 minutes |
|  |  | Step 5. Then peripheral stays in the list of discovered peripherals |
| 17383 | Discoverable peripheral with ModelId which matches ModelId filter is added to list of discovered peripherals | Step 1. Given ModelId filter is set to "AB3456,PS1234,CD7890" |
|  |  | Step 2. And application has started BLE discovery |
|  |  | Step 3. When a peripheral with ModelId "PS1234" becomes discoverable |
|  |  | Step 4. Then peripheral is added to list of discovered peripherals within 120 seconds |
| 17752 | Discoverable peripheral with ModelId which does not match ModelId filter is not added to list of discovered peripherals | Step 1. Given ModelId filter is set to "AB3456,CD7890,PS12" |
|  |  | Step 2. And application has started BLE discovery |
|  |  | Step 3. When a peripheral with ModelId "PS1234" becomes discoverable |
|  |  | Step 4. Then peripheral is not added to list of discovered peripherals |
| 14867 | Scenario: FW001 Perform a firmware upgrade for a valid upgrade package | Step 1. Given firmware upgrade push timeout is set to 30 seconds |
|  |  | Step 2. And application starts firmware upgrade push of Test.Latest.upg |
|  |  | Step 3. And firmware upgrade push is successful |
|  |  | Step 4. When firmware upgrade deployment timeout is set to 30 seconds |
|  |  | Step 5. And application starts firmware upgrade deployment |
|  |  | Step 6. Then firmware upgrade deployment is successful |
| 16934 | Scenario: FW007 Firmware Upgrade deployment times out | Step 1. Given firmware upgrade push timeout is set to 30 seconds |
|  |  | Step 2. And application start firmware upgrade push of Test.Latest.upg |
|  |  | Step 3. And firmware upgrade push is successful |
|  |  | Step 4. And firmware upgrade deployment timeout is set to 1 second |
|  |  | Step 5. And application starts firmware upgrade deployment |
|  |  | Step 6. When firmware upgrade deployment times out |
|  |  | Step 7. Then firmware upgrade deployment is unsuccessful |
|  |  | Step 8. And error message shows that 'deployment failed with timeout' |
| 14867 | Scenario: FW001 Perform a firmware upgrade for a valid upgrade package | Step 1. Given firmware upgrade push timeout is set to 30 seconds |
|  |  | Step 2. And application starts firmware upgrade push of Test.Latest.upg |
|  |  | Step 3. And firmware upgrade push is successful |
|  |  | Step 4. When firmware upgrade deployment timeout is set to 30 seconds |
|  |  | Step 5. And application starts firmware upgrade deployment |
|  |  | Step 6. Then firmware upgrade deployment is successful |
| 16933 | Scenario: FW006 Firmware Upgrade push times out | Step 1. Given firmware upgrade push timeout is set to 50 milliseconds |
|  |  | Step 2. And application starts firmware upgrade push of Test.Baseline.upg |
|  |  | Step 3. When firmware upgrade push timeout expires |
|  |  | Step 4. Then firmware upgrade push is unsuccessful |
|  |  | Step 5. And error message shows that 'push failed with timeout' |
| 14868 | Scenario: FW002 Cancel Firmware Upgrade push | Step 1. Given application starts firmware upgrade push of Test.Baseline.upg |
|  |  | Step 2. When firmware upgrade cancel timeout is set to 30 seconds |
|  |  | Step 3. And application cancels firmware upgrade push |
|  |  | Step 4. Then firmware upgrade push is stopped |
|  |  | Step 5. And application starts firmware upgrade push of Test.Latest.upg |
|  |  | Step 6. And firmware upgrade push is successful |
|  |  | Step 7. And application starts firmware upgrade deployment |
|  |  | Step 8. And firmware upgrade deployment is successful |
| 17847 | Scenario: FW009 Firmware Upgrade cancel times out | Step 1. Given firmware upgrade push timeout is set to 30 seconds |
|  |  | Step 2. And application starts firmware upgrade push of Test.Latest.upg |
|  |  | Step 3. And firmware upgrade cancel timeout is set to 10 milliseconds |
|  |  | Step 4. And application cancels firmware upgrade push |
|  |  | Step 5. When firmware upgrade cancel timeout expired |
|  |  | Step 6. Then error message shows that 'cancel failed with timeout' |
| 22751 | CppId of discovered peripheral is available | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. When a peripheral becomes discoverable |
|  |  | Step 4. Then CppId of discovered peripheral is shown in application |
| 22760 | ModelId of discovered peripheral, with a ModelId, is available | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. When a peripheral with a ModelId becomes discoverable |
|  |  | Step 4. Then ModelId of discovered peripheral is shown in application |
| 22764 | ModelId of discovered peripheral, without a ModelId, is not available | Step 1. Given application has started BLE discovery |
|  |  | Step 2. And modelId filter is empty |
|  |  | Step 3. When a peripheral without a ModelId becomes discoverable |
|  |  | Step 4. Then ModelId of discovered peripheral is not shown in application |
| 23990 | User pairing | Step 1. Given Mobile phone and Wifi Reference Node are connected to one Wifi network |
|  |  | Step 2. And commLib-demoapp is started |
|  |  | Step 3. And LAN scanning strategy is selected |
|  |  | Step 4. And Wifi Reference Node is discovered |
|  |  | Step 5. And Wifi Reference Node is selected |
|  |  | Step 6. When appliance-side pairing is requested with default values (as provided in the App) |
|  |  | Step 7. Then commLib-demoapp shows that pairing has "succeeded" |
|  |  | Step 8. And logging in device portal shows that AddRelationShipRequest results in '{ "RelationshipStatus": "pending" }' |
| 32880 | Firmware Port canUpgrade property is provided | Step 1. Given BLE Test App is running |
|  |  | Step 2. And a BLE Reference Node is discovered and selected |
|  |  | Step 3. And Firmware Upgrade screen is open |
|  |  | Step 4. When BLE Test App has received Firmware Port properties |
|  |  | Step 5. Then BLE Test App shows that value of "canUpgrade" property is "Yes" |
| 54008 | Trust certificate on first use | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And a clean install of commlib-demoapp is performed (no pin is stored) |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. When Device Name value is requested |
|  |  | Step 7. Then certificate is received |
|  |  | Step 8. And logging shows that pin is added for appliance |
|  |  | Step 9. And Device Name value is displayed |
| 54051 | Pin must be persisted for a connected managed appliance | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and a pin has already been stored for Wifi Reference Node |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. When Device Name value is requested |
|  |  | Step 7. And Device Name value is displayed |
|  |  | Step 8. And logging does not show that pin is added for appliance |
| 54418 | Pins must be persisted for multiple connected managed appliances | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and pins have already been stored for Wifi Reference Node 1 and Wifi Reference Node 2 |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And Wifi Reference Node 1 is discovered |
|  |  | Step 5. And Wifi Reference Node 1 is selected |
|  |  | Step 6. When Device Name value is requested |
|  |  | Step 7. And Device Name value is displayed |
|  |  | Step 8. And logging does not show that pin is stored |
|  |  | Step 9. And Wifi Reference Node 2 is discovered |
|  |  | Step 10. And Wifi Reference Node 2 is selected |
|  |  | Step 11. When Device Name value is requested |
|  |  | Step 12. And Device Name value is displayed |
|  |  | Step 13. And logging does not show that pin is added for appliance |
| 58371 | Allow pin mismatch to be resolved | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and a pin has already been stored for Wifi Reference Node |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. And factory reset procedure of Wifi Reference Node is performed (as described in attachment of this shared step) |
|  |  | Step 7. And Device Name value is requested |
|  |  | Step 8. And certificate is received |
|  |  | Step 9. And logging shows that pin mismatch is detected |
|  |  | Step 10. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 11. When user accepts new pin |
|  |  | Step 12. And Device Name value is requested |
|  |  | Step 13. Then Device name value is displayed |
|  |  | Step 14. And logging shows that re-pin occurred for appliance |
| 59923 | Don't allow pin mismatch to be resolved | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and a pin has already been stored for Wifi Reference Node |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. And factory reset procedure of Wifi Reference Node is performed (as described in attachment of this shared step) |
|  |  | Step 7. And Device Name value is requested |
|  |  | Step 8. And certificate is received |
|  |  | Step 9. And logging shows that pin mismatch is detected |
|  |  | Step 10. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 11. And user rejects new pin |
|  |  | Step 12. And appliance does not appear in "PIN MISMATCH" list |
|  |  | Step 13. And WiFi Reference Node is selected |
|  |  | Step 14. And Device Name value is requested |
|  |  | Step 15. And certificate is received |
|  |  | Step 16. And logging shows that pin mismatch is detected |
|  |  | Step 17. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 18. When user cancels new pin |
|  |  | Step 19. Then appliance appears in "PIN MISMATCH" list |
|  |  | Step 20. And WiFi Reference Node is selected |
|  |  | Step 21. And Device Name value is requested |
|  |  | Step 22. When Device Name value is requested |
|  |  | Step 23. Then certificate is received |
|  |  | Step 24. And logging shows that pin mismatch is detected |
|  |  | Step 25. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 26. When user accepts pin |
|  |  | Step 27. And Device Name value is requested |
|  |  | Step 28. And certificate is received |
|  |  | Step 29. And Device name value is received |
|  |  | Step 30. Then logging shows that pin is re-pinned |
| 58371 | Allow pin mismatch to be resolved | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and a pin has already been stored for Wifi Reference Node |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. And factory reset procedure of Wifi Reference Node is performed (as described in attachment of this shared step) |
|  |  | Step 7. And Device Name value is requested |
|  |  | Step 8. And certificate is received |
|  |  | Step 9. And logging shows that pin mismatch is detected |
|  |  | Step 10. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 11. When user accepts new pin |
|  |  | Step 12. And Device Name value is requested |
|  |  | Step 13. Then Device name value is displayed |
|  |  | Step 14. And logging shows that re-pin occurred for appliance |
| 59923 | Don't allow pin mismatch to be resolved | Step 1. Given Mobile phone and Wifi Reference Node are connected to the same Wifi network |
|  |  | Step 2. And commlib-demoapp is installed and a pin has already been stored for Wifi Reference Node |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And WiFi Reference Node is discovered |
|  |  | Step 5. And WiFi Reference Node is selected |
|  |  | Step 6. And factory reset procedure of Wifi Reference Node is performed (as described in attachment of this shared step) |
|  |  | Step 7. And Device Name value is requested |
|  |  | Step 8. And certificate is received |
|  |  | Step 9. And logging shows that pin mismatch is detected |
|  |  | Step 10. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 11. And user rejects new pin |
|  |  | Step 12. And appliance does not appear in "PIN MISMATCH" list |
|  |  | Step 13. And WiFi Reference Node is selected |
|  |  | Step 14. And Device Name value is requested |
|  |  | Step 15. And certificate is received |
|  |  | Step 16. And logging shows that pin mismatch is detected |
|  |  | Step 17. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 18. When user cancels new pin |
|  |  | Step 19. Then appliance appears in "PIN MISMATCH" list |
|  |  | Step 20. And WiFi Reference Node is selected |
|  |  | Step 21. And Device Name value is requested |
|  |  | Step 22. When Device Name value is requested |
|  |  | Step 23. Then certificate is received |
|  |  | Step 24. And logging shows that pin mismatch is detected |
|  |  | Step 25. And commlib-demoapp requests user to accept or to reject the new pin |
|  |  | Step 26. When user accepts pin |
|  |  | Step 27. And Device Name value is requested |
|  |  | Step 28. And certificate is received |
|  |  | Step 29. And Device name value is received |
|  |  | Step 30. Then logging shows that pin is re-pinned |
|  |  |  |
| 40231 | Use HTTPS to communicate over LAN with connectivity node which supports HTTPS | Step 1. Given Mobile phone and Wifi Reference Node are connected to one Wifi network |
|  |  | Step 2. And CommLib DemoApp is started |
|  |  | Step 3. And LAN scanning strategy is selected |
|  |  | Step 4. And Wifi Reference Node is discovered |
|  |  | Step 5. And Wifi Reference Node is selected |
|  |  | Step 6. When CommLib DemoApp reads port properties of device port |
|  |  | Step 7. Then logging shows that HTTPS is used as transfer protocol |
|  |  | Step 8. And logging shows that properties are successfully retrieved |
| 40236 | Use HTTPS to communicate over LAN with connectivity node which is not configured for a specific Hypertext Transfer Protocol | Step 1. Given Mobile phone and Airpurifier are connected to one Wifi network |
|  |  | Step 2. And CommLib Explorer App is started |
|  |  | Step 3. And LAN scanning strategy is selected |
|  |  | Step 4. And Airpurifier is discovered |
|  |  | Step 5. And Airpurifier is selected |
|  |  | Step 6. When CommLib Explorer App reads port properties of firmware port |
|  |  | Step 7. Then logging shows that HTTPS is used as transfer protocol |
|  |  | Step 8. And logging shows ECONNREFUSED (because AirPurifier does not support HTTPS) |
| 40416 | Upgrade DICommSampleApp when appliances are stored in persistent data | Step 1. Given Mobile phone, Airpurifier and Wifi Reference Node are connected to one Wifi network |
|  |  | Step 2. And DICommSampleApp from previous Platform release is installed and started |
|  |  | Step 3. And Wifi Reference Node and Airpurifier are discovered |
|  |  | Step 4. And Wifi Reference Node and Airpurifier are selected once and are available in list of managed devices |
|  |  | Step 5. When DICommSampleApp is upgraded to new release and started |
|  |  | Step 6. Then Wifi Reference Node and Airpurifier are in list of managed devices |
|  |  | Step 7. And DICommSampleApp can read Device name of Wifi Reference Node and Airpurifier |
| 40541 | Use HTTP to communicate over LAN with connectivity node which only supports HTTP | Step 1. Given Mobile phone and Airpurifier are connected to one Wifi network |
|  |  | Step 2. And CommLib DemoApp is started |
|  |  | Step 3. And LAN scanning strategy is selected |
|  |  | Step 4. And Airpurifier is discovered |
|  |  | Step 5. And Airpurifier is selected |
|  |  | Step 6. When Light state is modified |
|  |  | Step 7. Then logging shows that HTTP is used as transfer protocol |
|  |  | Step 8. And logging indicates that property was successfully set |
| 47366 | Use HTTPS to communicate over LAN with connectivity node which supports HTTPS | Step 1. Given Mobile phone and Wifi Reference Node are connected to one Wifi network |
|  |  | Step 2. And ddLogLevel in DiLocalConnection.m is set to DDLogLevelDebug |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And LAN scanning strategy is selected |
|  |  | Step 5. And Wifi Reference Node is discovered |
|  |  | Step 6. And Wifi Reference Node is selected |
|  |  | Step 7. When commlib-demoapp reads port properties of device port |
|  |  | Step 8. Then logging shows that HTTPS is used as transfer protocol |
| 47904 | Use HTTPS to communicate over LAN with connectivity node which is not configured for a specific Hypertext Transfer Protocol | Step 1. Given Mobile phone and Vacuum Cleaner are connected to one Wifi network |
|  |  | Step 2. And ddLogLevel in DILocalConnection.m is set to DDLogLevelDebug |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And Vacuum Cleaner is discovered |
|  |  | Step 5. And Vacuum Cleaner is selected |
|  |  | Step 6. When commlib-demoapp reads port properties of device port |
|  |  | Step 7. Then logging shows that HTTPS is used as transfer protocol |
|  |  | Step 8. And commlib-demoapp shows an error |
| 48062 | Use HTTP to communicate over LAN with connectivity node which only supports HTTP | Step 1. Given Mobile phone and Airpurifier are connected to one Wifi network |
|  |  | Step 2. And ddLogLevel in DILocalConnection.m is set to DDLogLevelDebug |
|  |  | Step 3. And commlib-demoapp is started |
|  |  | Step 4. And Airpurifier is discovered |
|  |  | Step 5. And Airpurifier is selected |
|  |  | Step 6. When Light state is modified |
|  |  | Step 7. Then logging shows that HTTP is used as transfer protocol |
| 49881 | CommLib listens for, receives and decrypts DIComm ChangeIndication UDP packets from the connectivity node when subscription is active | Step 1. Given Mobile phone and Wifi Reference Node are connected to one Wifi network |
|  |  | Step 2. And commLib-demoapp is started |
|  |  | Step 3. And Wifi Reference Node is discovered and selected |
|  |  | Step 4. When "Subscribe to time port" is enabled |
|  |  | Step 5. And "Enable Communication" is enabled |
|  |  | Step 6. Then time value is received continuously without errors |
| 50928 | CommLib listens for, receives and decrypts DIComm ChangeIndication UDP packets from a legacy connectivity node when subscription is active | Step 1. Given Mobile phone and an Air Purifier are connected to one Wifi network |
|  |  | Step 2. And commlib-demoapp is started |
|  |  | Step 3. And the Air Purifier is discovered and selected |
|  |  | Step 4. And "Subscribe to air port" is enabled |
|  |  | Step 5. And "Enable Communication" is enabled |
|  |  | Step 6. When pressing the light switch on the Air Purifier |
|  |  | Step 7. Then the light state is reflected in the app without errors |
| 17187 | Get value from time port | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And a BLE Reference Node is discovered and selected |
|  |  | Step 3. And "Stay connected" is disabled |
|  |  | Step 4. And "Subscribe" is disabled |
|  |  | Step 5. And "Continuously get Time" is disabled |
|  |  | Step 6. When application requests time value from time port |
|  |  | Step 7. Then time value is received without errors |
| 17188 | Get value from time port continuously, disconnect after request | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And a BLE reference Node is discovered and selected |
|  |  | Step 3. And "Stay connected" is disabled |
|  |  | Step 4. And "Subscribe" is disabled |
|  |  | Step 5. When application selects "continuously get time" |
|  |  | Step 6. And the application requests the time 50 times |
|  |  | Step 7. Then time value is received continuously without errors |
| 17189 | Get value from time port continuously, stay connected | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And a BLE Reference Node is discovered and selected |
|  |  | Step 3. And "Stay connected" is enabled |
|  |  | Step 4. When application selects "continuously get time" |
|  |  | Step 5. And the application requests the time 50 times |
|  |  | Step 6. Then time value is received continuously without errors |
| 17191 | Get value from time port continuously, via subscription/notification interface | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And BLE Reference Node is discovered and selected |
|  |  | Step 3. And "Stay connected" is disabled |
|  |  | Step 4. And "Continuously get Time" is disabled |
|  |  | Step 5. When "Subscribe" is enabled |
|  |  | Step 6. And the application receives the time 50 times |
|  |  | Step 7. Then time value is received continuously without errors |
| 17193 | Put value on device port | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And a BLE Reference Node is discovered and selected |
|  |  | Step 3. When Device Name value on device port is changed |
|  |  | Step 4. Then Device Name value is successfully changed |
| 17194 | Get value from time port continuously, communication error | Step 1. Given distance between phone and BLE Reference Node is 50 cm |
|  |  | Step 2. And a BLE Reference Node is discovered and selected |
|  |  | Step 3. And "Stay connected" is disabled |
|  |  | Step 4. And "Subscribe" is disabled |
|  |  | Step 5. And application selects "continuously get time" |
|  |  | Step 6. When BLE Reference Node is reset |
|  |  | Step 7. Then get time is unsuccessful |
|  |  | Step 8. And error message is displayed |
| 14869 | Scenario: FW003 Perform a firmware upgrade for an upgrade package with an invalid header version | Step 1. Given application starts firmware upload of Test.Product.InvalidHeaderVersion.upg |
|  |  | Step 2. And firmware upload is successful |
|  |  | Step 3. When application starts firmware deployment |
|  |  | Step 4. Then firmware deployment is unsuccessful |
|  |  | Step 5. And error message is displayed |
| 14870 | Scenario: FW004 Perform a firmware upgrade for an upgrade package with an invalid product | Step 1. Given application starts firmware upload of Test.Product.InvalidProduct.upg |
|  |  | Step 2. And firmware upload is successful |
|  |  | Step 3. When application starts firmware deployment |
|  |  | Step 4. Then firmware deployment is unsuccessful |
|  |  | Step 5. And error message is displayed |
| 14873 | Scenario: FW005 Perform a firmware upgrade for an upgrade package with an invalid signature | Step 1. Given application starts firmware upload of Test.Product.InvalidSignature.upg |
|  |  | Step 2. And firmware upload is successful |
|  |  | Step 3. When application starts firmware deployment |
|  |  | Step 4. Then firmware deployment is unsuccessful |
|  |  | Step 5. And error message is displayed |
| 17177 | Scenario: FW008 Communication error during firmware upload | Step 1. Given application starts firmware upload of Test.Latest.upg |
|  |  | Step 2. When BLE Reference Node is reset |
|  |  | Step 3. Then firmware upload is unsuccessful |
|  |  | Step 4. And error message is displayed |
| 18489 | Non-discoverable peripheral is removed from list of discovered peripherals, when peripheral is switched off | Step 1. Given application has started LAN discovery |
|  |  | Step 2. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 3. When peripheral becomes non-discoverable by switching it off |
|  |  | Step 4. Then peripheral is removed from the list of discovered peripherals within 120 seconds |
| 18492 | Non-discoverable peripheral is removed from list of discovered peripherals, when network is switched | Step 1. Given application has started LAN discovery |
|  |  | Step 2. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 3. When peripheral becomes non-discoverable by switching the network on the phone |
|  |  | Step 4. Then peripheral is removed from the list of discovered peripherals within 120 seconds |
| 18508 | Lost peripheral is rediscovered | Step 1. Given application has started LAN discovery |
|  |  | Step 2. And a peripheral is discovered and added to the list of discoverable peripherals |
|  |  | Step 3. And peripheral becomes non-discoverable by switching it off |
|  |  | Step 4. And peripheral is removed from the list of discovered peripherals |
|  |  | Step 5. When peripheral becomes discoverable by switching it on |
|  |  | Step 6. Then peripheral is added to the list of discoverable peripherals within 120 seconds |
| 18510 | Discoverable peripheral is added to list of discovered peripherals | Step 1. Given application has started LAN discovery |
|  |  | Step 2. When a peripheral becomes available |
|  |  | Step 3. Then peripheral is added to the list of discovered peripherals within 120 seconds |
| 18511 | Discoverable peripheral stays in list of discovered peripherals | Step 1. Given application has started LAN discovery |
|  |  | Step 2. And a peripheral is discovered and added to the list of discovered peripherals |
|  |  | Step 3. When peripheral remains discoverable for 2 minutes |
|  |  | Step 4. Then peripheral stays in the list of discovered peripherals |
| 16629 | A peripheral which has been connected once is added to persistent storage | Step 1. Given a mobile device with a test application |
|  |  | Step 2. And the discovered Airpurifier is not in the list of managed appliances |
|  |  | Step 3. And the mobile device is using connection "local" |
|  |  | Step 4. And the test application is used to toggle the Airpurifier light |
|  |  | Step 5. And the test application is killed |
|  |  | Step 6. When the test application is started |
|  |  | Step 7. Then the discovered Airpurifier is in the list of managed appliances |
| 31700 | CommLib upgrade from previous Platform release | Step 1. Given app (from previous Platform release) is installed and started on a mobile device |
|  |  | Step 2. And a discovered Airpurifier appliance |
|  |  | Step 3. And the mobile device is using connection "local" |
|  |  | Step 4. And app requests to pair with Airpurifier |
|  |  | Step 5. And the pairing request is successfull |
|  |  | Step 6. When app is upgraded to the new release |
|  |  | Step 7. And the mobile device is using connection "remote" |
|  |  | Step 8. Then the app can access the Airpurifier over the remote interface |
| 14726 | Scenario: PA001 Pairing happy path | Step 1. Given a mobile device |
|  |  | Step 2. And a discovered Airpurifier appliance |
|  |  | Step 3. And the app is "not paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection "local" |
|  |  | Step 5. When app requests to pair with Airpurifier |
|  |  | Step 6. Then the pairing request is successful within 30 seconds |
| 14734 | Scenario Outline: PA002 Pairing already paired | Step 1. Given a mobile device |
|  |  | Step 2. And a discovered Airpurifier appliance |
|  |  | Step 3. And the app is "paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection '@connection' |
|  |  | Step 5. When app requests to pair with Airpurifier |
|  |  | Step 6. Then the pairing request is successful within 30 seconds |
| 14743 | Scenario Outline: PA003 Cannot pair via wrong connection | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “not paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection '@connection' |
|  |  | Step 5. When app requests to pair with Airpurifier |
|  |  | Step 6. Then the pairing request is unsuccessful within 40 seconds |
| 14802 | Scenario: PA004 Cannot re-pair when there is no connection | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection 'none' |
|  |  | Step 5. When app requests to pair with Airpurifier |
|  |  | Step 6. Then the pairing request is unsuccessful within 40 seconds |
| 14803 | Scenario Outline: PA005 Unpair happy path | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection ‘@connection’ |
|  |  | Step 5. When app requests to unpair from Airpurifier |
|  |  | Step 6. Then the unpair request is successful within 40 seconds |
| 14805 | Scenario: PA006 Cannot unpair when there is no connection | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection ‘none’ |
|  |  | Step 5. When app requests to unpair from Airpurifier |
|  |  | Step 6. Then the unpair request is unsuccessful within 40 seconds |
| 14813 | Scenario: PA007 Cannot unpair when appliance is not paired and there is no connection | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “not paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection ‘none’ |
|  |  | Step 5. When app requests to unpair from Airpurifier |
|  |  | Step 6. Then the unpair request is unusccessful within 40 seconds |
| 14822 | Scenario Outline: PA008 Unpair when appliance is not paired | Step 1. Given a mobile device |
|  |  | Step 2. And an Airpurifier appliance |
|  |  | Step 3. And the app is “not paired" with the appliance |
|  |  | Step 4. And the mobile device is using connection ‘@connection’ |
|  |  | Step 5. When app requests to unpair from Airpurifier |
|  |  | Step 6. Then the unpair request is successful within 40 seconds |

## Automated Test Cases

The automated test cases are specified in a set of feature files inside Bitbucket in folder: [CommLib/commlib-bdd](https://bitbucket.atlas.philips.com/projects/COM/repos/commlib-bdd/browse).

To read the steps of a scenario, open its feature file in any text editor.

The feature files are listed in the table below:

|  |  |
| --- | --- |
| Feature Filename | Description |
| **put-prop.feature** | Scenarios to verify CommLib DiComm Put Proprties |
| **get-prop.feature** | Scenarios to verify CommLib DiComm Get Properties |
| **strategy-queueing.feature** | Scenarios to verify CommLib connection strategy request queueing |
| **availability.feature** | Scenarios to verify CommLib connection strategy availability |
| **discover-ble.feature** | Scenarios to verify CommLib BLE discovery |
| **discover-lan.feature** | Scenarios to verify CommLib LAN discovery |
| **subscription-notification.feature** | Scenarios to verify CommLib DiComm subscriptions and notifications |
| **https.feature** | Scenarios to verify CommLib https communication |
|  |  |

The table below shows all the test cases (scenarios) to verify the requirements of CommLib:

|  |  |
| --- | --- |
| Test Case ID | Test Case Title |
|  | Put Properties |
|  | CommBle00001-put-properties-req-happy-path |
|  | CommBle00002-put-properties-req-productid-with-multiple-digits |
|  | CommBle00003-put-properties-req-empty-port-identifier |
|  | CommBle00004-put-properties-req-null-data |
|  | CommBle00004a-put-properties-req-empty-data |
|  | CommBle00005-put-properties-req-empty-port-identifier-and-empty-data |
|  | CommBle00006-put-properties-req-nested-port-identifier |
|  | CommBle00007-put-properties-req-valid-json-types |
|  | CommBle00008-put-properties-req-valid-json-constructs |
|  | CommBle00009-put-properties-resp-happy-path-non-empty-json |
|  | CommBle00010-put-properties-resp-happy-path-empty-json |
|  | CommBle00011-put-properties-resp-not-allowed-empty-data |
|  | CommBle00012-put-properties-resp-error-no-such-port-and-not-allowed-empty-data |
|  | CommBle00012-put-properties-resp-error-no-such-property-and-json-object |
|  | CommBle00013-put-properties-resp-error-unknown-and-json-object |
|  | CommBle00014-put-properties-resp-error-out-of-memory-with-empty-json-object |
|  | CommBle00015-put-properties-resp-error-out-of-memory-with-non-json-but-still-ascii-data |
|  | CommBle00016-put-properties-resp-error-out-of-memory-with-non-json-and-non-ascii-data |
|  | CommBle00017-put-properties-resp-multiple-packets-7-chunks |
|  | CommBle00018-put-properties-resp-multiple-packets-3-chunks |
|  | CommBle00019-put-properties-resp-multiple-packets-2-chunks |
|  | CommBle00020-put-properties-resp-non-null-terminated-packet |
|  | CommBle00021-put-properties-resp-too-short-odd |
|  | CommBle00022-put-properties-resp-too-short-even |
|  | CommBle00023-put-properties-resp-excess-data-odd |
|  | CommBle00024-put-properties-resp-excess-data-even |
|  | CommBle00025-put-properties-req-max-size |
|  | CommBle00026-put-properties-req-max-size-plus-one |
|  | CommBle00027-put-properties-resp-max-size-payload |
|  | CommBle00028-put-properties-resp-max-size-payload-plus-1 |
|  | Get Properties |
|  | CommBle00101-get-properties-req-happy-path-connection-node |
|  | CommBle00102-get-properties-req-happy-path-non-connection-node |
|  | CommBle00103-get-properties-req-productid-with-multiple-digits |
|  | CommBle00104-get-properties-req-empty-port-identifier |
|  | CommBle00105-get-properties-req-nested-port-identifier |
|  | CommBle00106-get-properties-resp-happy-path-non-empty-data |
|  | CommBle00107-get-properties-resp-happy-path-empty-data |
|  | CommBle00108-get-properties-resp-leading-garbage-even |
|  | CommBle00109-get-properties-resp-leading-garbage-odd |
|  | CommBle00110-get-properties-resp-non-null-terminated-packet |
|  | CommBle00111-get-properties-resp-too-short-odd |
|  | CommBle00112-get-properties-resp-too-short-even |
|  | CommBle00113-get-properties-resp-excess-data-odd |
|  | CommBle00114-get-properties-resp-excess-data-even |
|  | CommBle00115-get-properties-resp-multiple-packets-2-chunks |
|  | CommBle00116-get-properties-resp-multiple-packets-3-chunks |
|  | CommBle00117-get-properties-resp-multiple-packets-5-chunks |
|  | CommBle00118-get-properties-resp-server-error-no-such-port-and-empty-data |
|  | CommBle00119-get-properties-resp-server-error-invalid-parameter-and-non-empty-data |
|  | CommBle00120-get-properties-resp-server-error-unknown-and-non-empty-data |
|  | CommBle00121-get-properties-resp-server-error-out-of-memory-with-only-json-data |
|  | CommBle00122-get-properties-resp-server-error-out-of-memory-with-empty-json-object |
|  | CommBle00123-get-properties-resp-max-size-payload |
|  | CommBle00124-get-properties-resp-max-size-payload-plus-1 |
|  | CommBle00125-get-properties-resp-valid-json-parsing |
|  | CommBle00126-get-properties-resp-valid-json-parsing-nested-objects |
|  | CommBle00127-get-properties-resp-no-json |
|  | CommBle00128-get-properties-resp-invalid-json |
|  | CommBle00129-get-properties-req-same-port-on-different-productids-is-allowed |
|  | Connection strategy queueing |
|  | CommBle00201-strategy-queue-depth1-happy-flow-1 |
|  | CommBle00202-strategy-queue-depth1-happy-flow-2 |
|  | CommBle00203-strategy-queue-depth1-timeout-of-first-req |
|  | CommBle00204-strategy-queue-depth1-unavailability |
|  | CommBle00205-strategy-queue-depth2-same-port |
|  | CommBle00206-strategy-queue-depth20-unique-ports |
|  | CommBle00207-strategy-queue-mixed-operations |
|  | Connection strategy availablility |
|  | Strategy is not available |
|  | Strategy is available |
|  | Strategy is available when device is found later |
|  | Strategy is not available, different device id |
|  | BLE Discovery |
|  | CommBle00301 CommLib starts discovery |
|  | CommBle00303 CommLib stops discovery |
|  | CommBle00304 CommLib stops discovery when not started, silent ignore |
|  | CommBle00305 CommLib discovers a supported appliance |
|  | CommBle00306 CommLib discovers multiple supported appliances of different type |
|  | CommBle00307 CommLib discovers multiple supported appliances of same type |
|  | CommBle00308 CommLib discovers only supported appliances |
|  | CommBle00309 CommLib discovers no supported appliances |
|  | CommBle00310 CommLib discovers appliance once |
|  | CommBle00311 CommLib discovers supported appliance after discovery was stopped |
|  | CommBle00312 CommLib discovers a supported appliance with a specific model id |
|  | CommBle00313 CommLib does not discover supported appliance with a specific model id |
|  | CommBle00314 CommLib discovers multiple supported appliances with a specific model id |
|  | CommBle00315 CommLib loses an appliance which was discovered, while discovery is active |
|  | CommBle00316 CommLib loses an appliance which was discovered, while discovery is inactive |
|  | CommBle00317 CommLib loses an appliance and rediscovers it |
|  | CommBle00318 CommLib loses multiple appliances that were discovered |
|  | LAN Discovery |
|  | CommLan00301 CommLib starts discovery |
|  | CommLan00302 CommLib stops discovery |
|  | CommLan00303 CommLib stops discovery when not started, silent ignore |
|  | CommLan00304 CommLib discovers a supported appliance |
|  | CommLan00305 CommLib discovers multiple supported appliances of different type |
|  | CommLan00306 CommLib discovers multiple supported appliances of same type |
|  | CommLan00307 CommLib discovers only supported appliances |
|  | CommLan00308 CommLib discovers no supported appliances |
|  | CommLan00309 CommLib discovers appliance once |
|  | CommLan00310 CommLib discovers supported appliance after discovery was stopped |
|  | CommLan00311 CommLib loses an appliance which was discovered, data in cache expires |
|  | CommLan00312 CommLib loses an appliance which was discovered, SSDP reports lost appliance |
|  | CommLan00313 CommLib loses an appliance which was discovered, network switch |
|  | CommLan00314 CommLib loses an appliance and rediscovers it |
|  | Subscriptions and notifications |
|  | Application subscribes to notifications on a port |
|  | Application unsubscribes from notifications on a port |
|  | Application receives a notification from a port |
|  | Application receives a notification from a port with an expired key |
|  | HTTPS |
|  | Uses HTTPS by default |
|  | Can be configured to use HTTP |
|  | Can be configured to not use HTTP |
|  | Discovery does not reconfigure to use HTTP |
|  | Discovery does not reconfigure to use HTTPS |

# Test Instructions

Not applicable. The test instructions are part of the steps of the test scenarios themselves.

# Revision History

| **Version** | **Date** | **Author** | **Description of Change** | **Reason for Change** |
| --- | --- | --- | --- | --- |
| 0.1 | 2017-May-01 | Gerard Arts | Initial draft | Creation |
| 1.0 | 2017-May-03 | Gerard Arts | Approver updated | Ready for Review |
| 1.1 | 2017-May-05 | Gerard Arts | Review remarks implemented | Review remarks implemented |
| 1.2 | 2017-May-10 | Bas Flaton | Removed test definitions that were meant for next release | Improvement |
| 2.0 | 2017-May-10 | Gerard Arts | Version changed | For approval |
| 2.1 | 2017-July-10 | Gerard Arts | Test cases added for PI 17.3. New test cases are related to HTTPS support . | Ready for Review |
| 2.2 | 2017-July-13 | Gerard Arts | Review remarks implemented | Review remarks implemented |

# Approval

| **Name** | **Role / Function** | **Date** (YYYY-MON-DD) | **Signature** |
| --- | --- | --- | --- |
| Matthijs Piek | Product Owner |  |  |