AllJoyn™ Service Framework Test Case Specification Template

Month Day, Year

Contents

[1 Introduction 3](#_Toc413670001)

[1.1 Purpose 3](#_Toc413670002)

[1.2 Scope 4](#_Toc413670003)

[1.3 References 4](#_Toc413670004)

[2 Environment setup 5](#_Toc413670005)

[2.1 Requirements 5](#_Toc413670006)

[2.2 Preconditions 5](#_Toc413670007)

[2.3 Test execution 5](#_Toc413670008)

[2.4 Parameters 5](#_Toc413670009)

[3 XXXX service framework test cases 7](#_Toc413670010)

[3.1 XXXX-vY-ZZ: [test case title] 7](#_Toc413670011)

[3.2 Template source file 8](#_Toc413670012)

[Appendix A Implementation Conformance Statement, ICS 9](#_Toc413670013)

[Appendix B Implementation eXtra Information for Testing, IXIT 10](#_Toc413670014)

[Appendix C Test Case Mapping Table, TCMT 11](#_Toc413670015)

# Introduction

[When naming the document, include the service framework name and version number, such as AllJoyn™ Configuration Service Framework 1.0 Test Case Specifications (see below for more details).

Chapter-level titles such as Introduction use the Heading 1 style. Purpose and Scope use the Heading 2 style. Body content uses the body style. Bullet-level content uses bullet lvl1, bullet lvl2, etc. This document supports four bullet levels.

XXXX refers to the name of the service framework. X.X refers to the version of service framework interface.

NOTE

At some point, the version number will be removed. The changes for each feature and service framework will be addressed per Alliance release (for example, 14.06, 14.10).

***Version Numbering:***

* *Core dependency: <core-year>.<core-month>, e.g. 14.06*
* *Feature release level: For use by services with more than one feature release built on the same core version*
* *Bug fix: versions not affecting incompatibility and certification*

*Append a letter for bug fix releases:*

*< core-year>.<core-month>.<feature><bugfix-letter>*

*Examples:*

* *14.06.00 is an initial service release using core 14.06*
* *14.06.00a is a bug fix release*
* *14.06.00b is another bug fix release*
* *14.06.01 is the next service feature release using core 14.06*

*As before, bug fix releases cannot add features, modify APIs, or introduce incompatible behavior. They are only intended to address critical stability and security issues.*

*The AllJoyn core will only add features on major releases 14.12, 15.04, etc.*

*Documentation should follow this same versioning.*

Red bracketed text is provided throughout the document to help create the document. Delete this text before contributing the document to the AllSeen Alliance.]

## Purpose

[If the About feature or XXXX service framework includes multiple interfaces, modify the end of the following statement to something like AllJoyn XXXX service framework X.X's collection of interfaces. If a service framework has distinct functions like the Notification service framework, list them and what they do.]

These test cases evaluate and verify the functionality related to the AllJoyn™ XXXX service framework exposed by a device through the XXXX X.X interface.

[Provide some context for the service framework, a brief description of what it does and if there is an application bundling requirement as with the Configuration service framework, or if there is something unique about the service framework.]

The About interface is required by an application to provide the discovery mechanism for the service framework interfaces that it supports, as well as providing the basic identification information.

## Scope

[The following is a blanket statement across all test case documents.]

These test cases are designed to determine if a device conforms to the XXXX X.X interface specifications. Successful completion of all test cases in this document does not guarantee that the tested device will interoperate with other devices.

## References

[The first document is the same across all test case documents. If you reference any RFCs or other documents, list them below. The RFC listed is only for reference for formatting purposes (provide the URL to the RFC).]

The following are reference documents.

* AllJoyn™ XXXX Service Framework X.X Interface Specification
* [RFC 2396 (Uniform Resource Identifiers (URI): Generic Syntax)](http://tools.ietf.org/html/rfc2396)

# Environment setup

## Requirements

[The following are listed across all test case documents. If your service framework supports distinct functions (for example, producer and consumer in the Notification service framework), list the roles the service framework supports.]

The following are required in order to execute these test cases:

* An AllJoyn-enabled device (the device under test or DUT) that supports the About feature 1.0
* A supported test device on which the test cases will run
* A Wi-Fi access point (referred to as the personal AP)

## Preconditions

Before running these test cases, it is assumed that:

* The DUT is connected to the personal AP
* The test device is connected to the personal AP
* At least one application on the DUT is announcing its capabilities through an About announcement

## Test execution

[The need for this section is based on the service framework's dependency on another service framework's interface such as the Config interface. The following example is from the Onboarding test case document. If this section isn't required, delete this section.]

NOTE

Some of the test cases involve interacting with the Config 1.0 interface (if supported by the DUT) and running through scenarios where the passcode is changed.

If the DUT does not conform to the interface specification or if there is an issue that causes a test case step to fail, manual steps may be necessary to return the DUT to the proper state (for example, passcode set to the default passcode) before testing can continue.

## Parameters

[The following parameters are listed across all test case documents. If other parameters must be identified, list and define.

The table uses the table heading style for the heading. The table content uses the table entry style.]

Table 1. Parameters for the XXXX service framework

| Parameter | Description |
| --- | --- |
| DeviceId | Device ID of the DUT |
| AppId | Application ID of the application on the DUT |

# XXXX service framework test cases

[Test cases are typically listed in the same sequence as the interface(s), methods, and parameters defined in the service framework's Interface Specification document. But t*he test case execution order can be different than the numerical sequence implied by the test case number*. *Ideally, test cases should be defined to run in the order of definition/numbering, but a different ordering can be specified when writing the test cases.*

If, after you create test cases and execute them during the testing phase, you need to delete a test case created in error, there is no need to re-sequence the test case numbers. *This is more from a consistency standpoint.*

To ensure the developers have a complete picture of what the test cases are testing for a given function, review other test case documents for the level of detail provided.]

## XXXX-vY-ZZ: [test case title]

[Y in the test case title represents the version of the service framework’s interface (for example, v1).

ZZ represents the numerical sequence of the test cases, for example, 01, 02, 03.

Here is a sample test case title: Config-v1-01: System App AppId equals DeviceId.

The bold titles in each section (Objective, Procedure, and Expected results) use the subhead indented style. Each test case has the same three titles. It is recommended that you build out a test case and then copy and paste it for use as a base for the subsequent test cases.]

Objective

[If a test case is considered a prerequisite or postrequisite to running any other, state "This test case must be executed first before running other XXXX test cases” or “This test case must be executed last after running other XXXX test cases” or “This test case must be executed before running XXXX test case" or “This test case must be executed after running XXXX test case.”

Complete the statement below based on the function the test case is verifying.]

Verify that the test device can XXXX….

For example:

Verify whether the AppId of the DUT’s System App (the application supporting the Config interface) matches the DUT’s DeviceId.

Procedure

[The first step is used across all test case documents. Step 2 is also used across all documents but what happens after receiving the announcement varies for each service framework. The last step is always the same. If you are testing a function that returns an error or the DUT doesn't support an interface specified in the Test execution section, state that the test case exits at that point in the procedure.

The style for numbering the steps is called numbrd list. If a step requires bullet points use the bullet lvl2 style. If a step requires sequential sub-steps (a, b, c), use numbrd list +.]

1. The test device listens for an About announcement from the application on the DUT.
2. After receiving an About announcement from the application, the test device….
3. [Detail the additional steps that support the function your service framework has.]
4. The test device leaves the session.

Expected results

[The expected results mirror the procedure's steps and provide detailed information for a given step such as listing what a method call returns, verifying the property values being returned, or which fields are returned for a parameter.

The first bullet point is the same across all test case documents.]

* The test device receives an About announcement from the application on the DUT.
* [Detailed results for the function being tested.]

## Template source file

The Word document to create this template file is provided below. Delete this section after downloading the document.



Implementation Conformance Statement, ICS

[This appendix will include a collection of one or more tables that will list the features that a product may support related to the Service Framework to be tested.]

**Table A.1: Table of optional features 1**

| Item | Description | Status | Support | Mnemonic |
| --- | --- | --- | --- | --- |
| 1 | Support of Feature 1.1 | [status 1.1] |  | [Mnemonic 1] |
| 2 | Support of Feature 1.2 | [status 1.2] |  | [Mnemonic 2] |
| ….. |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Status 1.1 Condition 1.1  Status 1.2 Condition 1.2  … …. | | | | |

**Table A.2: Table of optional features 2**

| Item | Description | Status | Support (Yes/No) | Mnemonic |
| --- | --- | --- | --- | --- |
| 1 | Support of Feature 2.1 | [status 2.1] |  | [Mnemonic 1] |
| 2 | Support of Feature 2.2 | [status 2.2] |  | [Mnemonic 2] |
| ….. |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Status 2.1 Condition 2.1  Status 2.2 Condition 2.2  … …. | | | | |

[The table below shows a partial example for lighting service ICS]

**Table A.1: Lighting features**

| Item | Description | Status | Support (Yes/No) | Mnemonic |
| --- | --- | --- | --- | --- |
| 1 | Support of Lighting Service Framework | O |  | ICSL\_LightingServiceFramework |
| 2 | Support of LampService Interface | C.001 |  | ICSL\_LampServiceInterface |
| 3 | Support of LampParameters Interface | C.001 |  | ICSL\_LampParametersInterface |
| 4 | Support of LampDetails Interface | C.001 |  | ICSL\_LampDetailsInterface |
| 5 | Support of light dimming | C.002 |  | ICSL\_Dimmable |
| 6 | Support of variable color | C.002 |  | ICSL\_Color |
| 7 | Support of variable color temperature | C.002 |  | ICSL\_ColorTemperature |
| 8 | Support of effects | C.002 |  | ICSL\_Effects |
| 9 | Support of LampState Interface | C.001 |  | ICSL\_LampStateInterface |
| C.001 IF (L.1/1) THEN M ELSE N/A  C.002 IF (L.1/4) THEN O ELSE N/A | | | | |

Implementation eXtra Information for Testing, IXIT

[This appendix will include a collection of one or more tables that will list the extra Information for testing that a product may require to be tested]

**Table A.1: Table of optional features 1**

| Item | Name | Description | Default Value | Mnemonic | Value |
| --- | --- | --- | --- | --- | --- |
| 1 | Extra Info for testing. Element 1 |  |  | [Mnemonic 1] |  |
| 2 | Extra Info for testing. Element 2 |  |  | [Mnemonic 2] |  |
| ….. |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

[The table below shows a partial example for lighting service IXIT]

**Table A.1: Lighting features**

| Item | Name | Description | Default Value | Mnemonic | Value |
| --- | --- | --- | --- | --- | --- |
| 1 | LampService Version | LampService Interface version number | 1 | IXITL\_LampServiceVersion |  |
| 2 | LampParameters Version | LampParameters Interface version number | 1 | IXITL\_LampParametersVersion |  |
| 3 | LampDetails Version | LampDetails Interface version number | 1 | IXITL\_LampDetailsVersion |  |
| 4 | LampState Version | LampState Interface version number | 1 | IXITL\_LampStateVersion |  |

Test Case Mapping Table, TCMT

The applicability of each individual test is identified in the table B.1.

The applicability of every test is formally expressed by the use of Boolean expression that are based on parameters (ICS) included in annex A of the present document.

The columns in Table B.1 have the following meaning:

*Test case column*

The Test case column indicates the test case number for each test case as described in the Control Panel Service Framework test case specification for which the applicability is identified.

*Description column*

The Title column indicates the title of each test case as described in the Control Panel Service Framework test case specification for which the applicability is identified.

*Release column*

The Release column indicates the earliest release from which each test case is applicable, except if otherwise stated of an individual test case.

*Status column*

The following notations are used for the Status column:

A applicable - the test is applicable.

O optional – the capability may be supported or not.

N/A not applicable – in the given context, the test case is not applicable.

Ci conditional – the test is applicable ("A") or not ("N/A") depending on the support of other optional or conditional items. "i" is an integer identifying an unique conditional status expression which is defined immediately following the table. For nested conditional expressions, the syntax "IF ... THEN (IF ... THEN ... ELSE...) ELSE ..." is used to avoid ambiguities.

[This appendix will include a table (B.1) defining the test cases applicability according to the definitions detailed above. ]

Table B.1: Test Case Mapping Table

| Test case | Description | Release | Status |
| --- | --- | --- | --- |
| Test case 1 | Description of test case 1 | Release of test case 1 | Status 1 |
| Test case 2 | Description of test case 2 | Release of test case 2 | Status 2 |
| Test case 3 | Description of test case 2 | Release of test case 3 | Status 3 |
| …. |  |  |  |
|  |  |  |  |
| Status 1 Condition 1  Status 2 Condition 2  Status 3 Condition 3  …. | | | |

[The table below shows a partial Lighting Test Case Mapping Table example]

Table B.1: Test Case Mapping Table

| Test case | Description | Release | Status |
| --- | --- | --- | --- |
| LSF\_Lamp-v1-01 | Service Interface Version equals 1 | 14.06 | C01 |
| LSF\_Lamp-v1-02 | Lamp Service Version equals 1 | 14.06 | C02 |
| …. |  |  |  |
| C01 IF (L.1/1) THEN A ELSE N/A  C02 IF (L.1/2) THEN A ELSE N/A | | | |