SCA 4.1 Applications Verification Plan

Document WINNF-TS-4002

Version V1.0.0
18 December 2018

TERMS, CONDITIONS & NOTICES

This document has been prepared by the Work Group of WInnF Project SCA-2017-001 "Verification of SCA 4.1 Applications" to assist The Software Defined Radio Forum Inc. (or its successors or assigns, hereafter "the Forum"). It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the SCA Test and Evaluation Work Group.

Contributors to this document that have submitted copyrighted materials (the Submission) to the Forum for use in this document retain copyright ownership of their original work, while at the same time granting the Forum a non-exclusive, irrevocable, worldwide, perpetual, royalty-free license under the Submitter's copyrights in the Submission to reproduce, distribute, publish, display, perform, and create derivative works of the Submission based on that original work for the purpose of developing this document under the Forum's own copyright.

Permission is granted to the Forum's participants to copy any portion of this document for legitimate purposes of the Forum. Copying for monetary gain or for other non-Forum related purposes is prohibited.

THIS DOCUMENT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS DOCUMENT.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

This document was developed following the Forum's policy on restricted or controlled information (Policy 009) to ensure that that the document can be shared openly with other member organizations around the world. Additional Information on this policy can be found here: http://www.wirelessinnovation.org/page/Policies_and_Procedures

Although this document contains no restricted or controlled information, the specific implementation of concepts contain herein may be controlled under the laws of the country of origin for that implementation. Readers are encouraged, therefore, to consult with a cognizant authority prior to any further development.

Wireless Innovation Forum ™ and SDR Forum ™ are trademarks of the Software Defined Radio Forum Inc.

Table of Contents

List of Tables

Contributors

The following individuals and their organization of affiliation are credited as Contributors to development of the specification, for having been involved in the work group that developed the draft then approved by WInnF member organizations:

- Ed Brabant, JTEL
- Eric Christensen, JTNC Standards
- Huan Dao, JTEL
- Jean-Philippe Delahaye, DGA
- James Evangelos, JTNC Standards
- James Ezick, Reservoir Labs
- Chris Hagen, Rockwell Collins
- François Lévesque, NordiaSoft
- Charles Linn, Harris
- Jimmie Marks, Raytheon
- Eric Nicollet, Thales Communication & Security
- Kevin Richardson, JTNC Standards
- Sarvpreet Singh, Fraunhofer FKIE
- Jonathan Springer, Reservoir Labs

SCA 4.1 Applications Verification Plan

1 Introduction

The purpose of this document is to define a plan on how the verification of SCA 4.1 applications would be performed. This would determine if such applications can be considered as compliant to the SCA 4.1 specification. In order to determine if an application is compliant, it will have to be verified by a set of verification procedures. No specific assumption is made regarding the nature of the designated verification authority for the establishment of the verification plan. The plan also aims to help in cutting down costs and shorten schedules for SCA 4.1 verification.

The Application Verification Plan will define:

- What would be the elements under verification for an application?
- What are the SCA requirements that are applicable to an application to determine its compliancy to the SCA standard?
- What are the verification methods that can be used to define the verification procedures?
- What is the strategy that will be employed to determine the verification method(s) for verifying a specific requirement?

The verification procedures developed in accordance with this plan will be programming language independent, therefore they will be applicable for any SCA 4.1 based product. The target objective will be an end state where the same method or product will be able to execute the specified procedures. However, if there are any programming language or middleware specializations, they will be provided in the test procedure implementation.

The following terms are used within this document and should be interpreted as described in RFC-2119:

- SHALL is a mandatory requirement (negative is SHALL NOT)
- SHOULD is recommended requirement/best practice (negative is SHOULD NOT)
- MAY is an optional requirement, i.e., something that is allowed (negative is NEED NOT)

2 Verification Approach

The purpose of the verification is to verify the compliance of a complete “SCA 4.1 Application” submitted by an application “Provider” to a conformance “Tester”. Earlier tests conducted by the “Provider” in the course of the product development are not in the scope of this document.

SCA 4.1 Application verification can be performed at either the source code (does the code comply with the standardized syntactic requirements) or executable code (does the application behave in accordance with the standardized component semantics) level. The appropriate method for each requirement is dictated by the requirement text and other factors such as cost, time and resources that can be applied. Once verified, there is an implicit guarantee that an SCA compliant Operating Environment can operate or manage the Application although additional porting may be required.

"Provider", "User" and "Tester" are roles that might be undertaken by any actors of the ecosystem (e.g. industry, government, etc.) depending on the business context where the verification is taking place.

As a precondition for verification to take place, the "Provider" shall provide:

- The technical package corresponding to the application under verification (composed of the elements described in Section 3).
- The conformance claim related to the submitted application that at least identifies the profiles and/or unit of functionalities of the SCA 4.1 Standard against which conformance is expected.

The conformance claim, being based on unit of functionalities, implicitly provide the list of SCA requirements (as defined in section 4) that must be verified to establish the conformance. For each requirement, one or more verification procedure(s) will be defined by following the guidelines described in this document.

The outcome of the verification is a statement issued by the "Tester" based on the result of each verification procedure which are exercised according to the conformance claim of the submitted technical package.

3 Elements under Verification

An SCA application is an assembly of one or more components ("software modules") that are connected to perform a certain task. The assembly must provide a meta-data file to describe itself. The individual components composing the assembly must also provide one or more meta-data files each along with the binary file and any other file(s) required for their execution. For verification(s) involving only the execution of the assembly these elements should be sufficient to proceed. However, when other verification means must be used as described in Section 4, more elements would be needed to perform the verifications. Therefore, an application under verification shall provide:

- **Metadata**: The Domain Profile XML files of the application assembly and all its components that are necessary for the execution of the application by an SCA Core Framework (CF);
- **Runtime code**: The binary file(s) in the appropriate format that are necessary for the execution of the application by an SCA Core Framework (CF);
- **Source code**: All source code files written by the application developer along with the source code files of all third-party software composing the application for which the corresponding runtime code is not included in the operating environment (OE) on which the verification is performed;
- **Build files**: The files (make files, scripts, tool chain specific files, etc.) used to produce the runtime code of the application.
- **Production assumptions**: Documentation describing assumptions that are made for the production of the application (specific configuration in the development, name and version of the development tools (tool chain) used to produce the application, etc.).

- **Runtime assumptions**: Documentation describing assumptions that are made for the execution of the application (specific configuration in the OE, etc.).

4 Application Requirements

The Table 1 and Table 2 lists all the requirements to which an SCA 4.1 application shall be compliant [Ref2]. The requirements which are always mandatory for components of an application are listed in Table 1. However, some requirements which are applicable only if the application (or its components) implements certain Unit of Functionalities (UoFs) [Ref3] are listed in Table 2. The column "Applicable SCA Component(s) indicates to which component a requirement is applicable. The column "Applicable UoF" indicates "None" when a requirement is always applicable to a component and not only for one or more specific UoF; otherwise it indicates a specific UoF that must be implemented for the requirement to be applicable. The column SCA 2.2.2 indicates whether or not a requirement was also enforced in the version 2.2.2 of the SCA (this column is present only for information purpose and has no consideration for the verification purpose).

Table 1: SCA 4.1 Requirements applicable to an Application that are mandatory

Requirement	Requirement Allocation	Applicable SCA Component(s)	Applicable UoF	SCA 2.2.2
SCA386	Both	BaseFactoryComponent	None	X
SCA387	Both	BaseFactoryComponent	None	X
SCA388	Both	BaseFactoryComponent	None	X
SCA389	Both	BaseFactoryComponent	None	X
SCA427	Both	BaseComponent	None	
SCA430	Both	BaseComponent	None	
SCA548	Both	BaseComponent	None	
SCA540	Both	BaseFactoryComponent	None	X
SCA413	Both	BaseFactoryComponent	None	
SCA414	Both	BaseFactoryComponent	None	
SCA549	Both	BaseFactoryComponent	None	
SCA169	AP	ManageableApplicationComponent	None	X
SCA173[1]	AP	ApplicationComponent	AEP Compliant	X
SCA457	AP	ApplicationComponent	None	X
SCA551	AP	ApplicationComponent	None	
SCA455	AP	ManageableApplicationComponent	None	X
SCA456	AP	ManageableApplicationComponent	None	X
SCA520	AP	ManageableApplicationComponent	None	
SCA166	AP	ManageableApplicationComponent	None	X
SCA167	AP	ManageableApplicationComponent	None	X

[1] Applicable to AEP Compliant UoF which is always mandatory for components of an application.

SCA550	AP	ManageableApplicationComponent	None	
SCA175	AP	ApplicationControllerComponent	None	
SCA176	AP	ApplicationControllerComponent	None	
SCA415	AP	ApplicationComponentFactoryComponent	None	
SCA521	AP	ApplicationComponentFactoryComponent	None	
SCA522	AP	ApplicationComponentFactoryComponent	None	
SCA155	AP	AssemblyComponent	None	X
SCA156	AP	AssemblyComponent	None	
SCA463	Both	BaseComponent	None	X
SCA471[2]	AP	OS	AEP Provider	X
SCA501	Both	BaseComponent	None	X
SCA502	Both	BaseComponent	None	X
SCA496	AP	ApplicationControllerComponent	None	X
SCA503	Both	BaseComponent	None	X
SCA494	Both	BaseComponent	None	X
SCA495	Both	BaseComponent	None	X

Table 2: SCA 4.1 Requirements applicable to an Application that are associated to a UoF

Requirement	Requirement Allocation	Applicable SCA Component(s)	Applicable UoF	SCA 2.2.2
SCA420	Both	BaseComponent	Log Producer, Configurable	X
SCA421	Both	BaseComponent	Log Producer	X
SCA423	Both	BaseComponent	Log Producer	X
SCA429	Both	BaseComponent	Configurable	X
SCA545	Both	BaseComponent	Configurable	
SCA26	Both	BaseComponent	Configurable	X
SCA27	Both	BaseComponent	Configurable	X
SCA28	Both	BaseComponent	Configurable	X
SCA29	Both	BaseComponent	Configurable	X
SCA30	Both	BaseComponent	Configurable	X
SCA31	Both	BaseComponent	Configurable	X
SCA432	Both	BaseComponent	LifeCycle	
SCA15	Both	BaseComponent	LifeCycle	X
SCA518	Both	BaseComponent	Releasable	X
SCA574	Both	BaseFactoryComponent	Releasable	
SCA16	Both	BaseComponent	Releasable	X
SCA17	Both	BaseComponent	Releasable	X
SCA18	Both	BaseComponent	Releasable	X

[2] Applicable to AEP Provider (AEP Compliant) UoF which is always mandatory for components of an application.

SCA433	Both	BaseComponent	Controllable	
SCA32	Both	BaseComponent	Controllable	
SCA33	Both	BaseComponent	Controllable	
SCA34	Both	BaseComponent	Controllable	X
SCA36	Both	BaseComponent	Controllable	
SCA37	Both	BaseComponent	Controllable	X
SCA547	Both	BaseComponent	Connectable	
SCA7	Both	BaseComponent	Connectable	X
SCA519	Both	BaseComponent	Connectable	
SCA8	Both	BaseComponent	Connectable	X
SCA10	Both	BaseComponent	Connectable	X
SCA11	Both	BaseComponent	Connectable	
SCA12	Both	BaseComponent	Connectable	X
SCA13	Both	BaseComponent	Connectable	X
SCA14	Both	BaseComponent	Connectable	
SCA82	AP	ApplicationComponent	Component Registration	
SCA424	Both	BaseComponent	Event Producer	X
SCA425	Both	BaseComponent	Event Producer	X
SCA444	Both	BaseComponent	Event Consumer	X
SCA426	Both	BaseComponent	Interrogable	
SCA541	Both	BaseFactoryComponent	Interrogable	X
SCA6	Both	BaseComponent	Interrogable	X
SCA168	AP	ManageableApplicationComponent	Interrogable	X
SCA428	Both	BaseComponent	Testable	X
SCA546	Both	BaseComponent	Testable	
SCA19	Both	BaseComponent	Testable	X
SCA21	Both	BaseComponent	Testable	X
SCA23	Both	BaseComponent	Testable	X
SCA24	Both	BaseComponent	Testable	X
SCA25	Both	BaseComponent	Testable	X
SCA500	AP	ApplicationControllerComponent	Channel Extension	X
SCA506	AP	ApplicationComponent	CORBA Provider	X

The verification procedures for the requirements that are part of the AEP Compliant (and AEP Provider) UoF would have to take into consideration the SCA Application Environment Profile (AEP, LwAEP or ULwAEP) selected for an application. Along the same lines, the verification procedures for the requirements that are part of the CORBA Provider UoF would have to take into consideration the SCA CORBA Profile (Full, Lightweight or Ultra-Lightweight) selected for an application.

5 Verification Methods Definition

The verification procedures that would be defined for the requirements listed in Table 1 and Table 2 shall use one or more of the following verification methods. Some methods can have a level of automation as defined in [Ref1].

Inspection: Visual inspection of equipment and evaluation of drawings and other pertinent design data and processes should be used to verify conformance with characteristics such as physical, material, part, and product marking and workmanship.

For the SCA application, this would imply using a text editor or a software Integrated Development Environment (IDE) application to search the source files, to locate executable statements which implement the behavior of the SCA requirement(s) applicable to the test case. A test engineer will analyze the source statements identified to determine whether those statements comply with the behavior(s) described by SCA requirement(s) applicable to the test case.

Analysis: Analysis is the use of recognized analytic techniques (including computer models) to interpret or explain the behavior/performance of the system element. Analysis of test data or review and analysis of design data should be used as appropriate to verify requirements.

If the analysis can be performed by a tool or a combination of tools in an automated way, the method shall be Automated Analysis and the list of tools shall be provided.

For an SCA application, this could imply using static analysis for an application which implements logic to parse the source code and SCA Domain Profile XML files to identify statements which implement the behavior of the SCA requirement(s) applicable to the test case. It could also imply using a linker tool to report success or failure of source code linkage against a library containing or not containing well-known API.

If the analysis can be partially automated and partially done manually (before and/or after the automated part) then the method shall be Partial Automated Analysis, otherwise Manual Analysis.

Demonstration: Demonstration is the performance of operations at the system or system element level where visual observations are the primary means of verification. Demonstration is used when quantitative assurance is not required for verification of the requirements.

Test: Test is an activity designed to provide data on functional features and equipment operation under fully controlled and traceable conditions. The data are subsequently used to evaluate quantitative characteristics.

For the SCA Application, a runtime SCA test tool would invoke functions (including CORBA operations) specified by the SCA requirement(s) applicable to the test case. The runtime SCA test tool can determine from the output of those functions or via a query method of the results of those functions whether the function complies with the behavior(s) described by SCA requirement(s) applicable to the test case. The result of a test shall be a true positive or a true negative.

6 Verification Result Category Definition

Each verification procedure that will be defined shall provide as output a result from one of the following terms as defined in [Ref1]:

- **Necessary**: A necessary result means that the requirement does not hold if the test fails. A result of pass does not mean that the requirement is verified, it only indicates that proof of non-compliance could not be found. A necessary test may falsely indicate a pass when the result should be fail.
- **Sufficient**: A sufficient result means that the requirement holds if the test passes. A result of fail does not mean that the requirement is not satisfied, it only means that a proof of compliance could not be found. Sufficient tests are often coupled with pre-conditions on how the product must be developed (e.g., a coding standard that makes certain properties transparent). A sufficient test may falsely indicate a fail when the result should be a pass.
- **Neither**: A test that is neither necessary nor sufficient produces results that must be manually post-processed to make a pass/fail determination. These tests are capable of generating both false-positive and false-negative results. Often, a "neither" test does not attempt to provide any determination of compliance and only collects information for further inspection or analysis.

Note that the term "Both" is also used in [Ref1] to indicate a result that can be "Necessary" and "Sufficient". While such a result will be possible as an output result of a verification procedure, the term "Both" will not be used. Instead, the explicit terms "Necessary and Sufficient" will be used to indicate such result.

7 Verification Methods Assignment

The verification method(s) to assign for the verification of a requirement should be based on cost (recurring and non-recurring), time, reliability (reproducible vs interpretation), accuracy, etc. The less expensive and reproducible method should always be preferred.

The following presents a list of verification methods:

- Inspection
- Analysis
- Demonstration
- Test

The order of precedence of the above methods would be decided on a case by case basis for each requirement. In addition, to provide reproducibility and remove misleading interpretation of the requirement verification procedure, automated verification methods would be preferred over manual methods.

For each verification procedure there must be defined pre and post conditions. Also, each requirement will have a preferred verification method which meets the goals of least expensive and most accurate. If the preconditions for the preferred verification method are not met, an

alternative verification method may also be provided. A single verification method may be insufficient for complex requirements.

8 References

[Ref0] : Software Communications Architecture Specification, Version 4.1, 20 August 2015
[Ref1] : SCA Test, Evaluation, and Certification, WInnF-12-P-0005, V1.0.0, 25 July 2016
[Ref2] : SCA 4.1 Requirements Allocation, Objectives and Verification Criteria, Working Document WINNF-16-P-0025-V1.0.0, 17 March 2017
[Ref3] : SCA 4.1 Appendix F – Attachment 1: SCA Conformance Mapping