Proposal for SCA v4.1
SCA Clarifications

Document WINNF-15-R-0028

Version V1.0.0

May 29, 2015

Terms and Conditions

This document has been prepared by the WInnF SCA 4.1 Draft Issue Adjudication WG 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 It may be amended or withdrawn at a later time and it is not binding on any member of the Forum.

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.

Intellectual Property Rights

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.

Proposal

This document contains a proposal to change the Draft SCAv4.1 specification to provide clarification for the following submitted issues:

- Issue #123, 127 Naming Service Compatibility
- Issue #154, 156 IDL to C++11 Type Mapping
- Issue #166, Factory Release Component

Proposal author:
Name: Jimmie Marks
Organization: Raytheon
Address: 1010 Production Road, Fort Wayne IN 46808
Phone number: 260-429-6422
email: Jimmie_T_Marks@raytheon.com

Proposal contributors / reviewers:

- François Lévesque, NordiaSoft
- Gerald Bickle, Raytheon
- Ken Dingman, Harris
- Hugues Latour, CRC
- Kevin Richardson, JTNC
- Eric Christensen, JTNC

Recommendation

Issue #123, 127 Naming Service Compatibility

Issue Description:

In the webinar it was mentioned that the "Naming Service has been removed," but it still appears to be a requirement in the 4.1 draft in a number of places. Its spec is a Normative Reference and there are requirements in the text such as in this paragraph: Upon execution of a software module by the create operation, a Resource or a ResourceFactory component shall register with the Naming Service." and the next paragraph seems to allow ...via the CORBA Naming Service or a resource factory. "but the next sentence says: "The ResourceFactory object reference is obtained by using the CORBA Naming Service."

So it still appears to be a requirement.

Summary of the Proposal

Summary of the Proposal:
Move the backward compatibility text (i.e. Backwards Compatible Alternative Requirements sections, etc.) from the main speciation document to the Appendix F. Created new section F6.5 and create new Appendix F6.5 sub-sections.

Detailed Proposal :
<u>Move</u>

- Section 3.1.2, Transfer Mechanism & Services
  SCA544 The OE shall provide a naming capability which implements the CosNaming module NamingContext interface operations: bind, bind_new_context, unbind, destroy, and resolve as defined in the OMG Naming Service Specification [4] using the IDL found in Appendix A of that reference.
- Section 3.1.3.3.1.1.5.1.6 Backwards Compatible Alternative Requirements
  All text
- Section 3.1.3.3.1.3.5.1.6 Backwards Compatible Alternative Requirements
  All text, Remove N/A(s)
- Section 3.1.3.3.1.5.5.1.5 Exceptions/Errors
  SCA552 The installApplication operation shall raise the ApplicationInstallationError exception when SCA V2.2.2 application installation is not supported.
- Section 3.1.3.3.2.1.5 Backwards Compatible Alternative Requirements
  All text. Remove N/A(s)
- Section 3.1.3.3.2.2.3 Semantics
  SCA555 The create operation shall instantiate a SCA V2.2.2 compliant application if the SAD does not have a sca_version element. The create operation maps the SCA V2.2.2 application values into the created ApplicationManagerComponent.
- Section 3.1.3.3.2.2.5 Backwards Compatible Alternative Requirements
  All text. Remove N/A(s)
- Section 3.1.3.3.2.3.3 Semantics
  SCA558 The installApplication operation shall install a SCA V2.2.2 [3] compliant application.

Continued on next page

Summary of the Proposal (cont)

To:
New section - F.6.5 Application Backward Compatible Units of Functionality
New Text - Following sections provide the alternative V2.2.2 requirements to provide capability to manage SCA V2.2.2 compliant applications.

Create (new) subsections for moved sections above. Reference appropriate SCA 4.1 sections within the new sections.

Proposed new subsections. The titles of new sub-sections has flexibility other than proposed.

F.6.5.1 Naming Service
F.6.5.2 ApplicationManager ReleaseOjbect
F.6.5.3 ApplicationManager Create
F.6.5.4 InstallApplication Exception/Errors
F.6.5.5 ApplicationManager
F.6.5.6 ApplicationFactoryComponent
F.6.5.7 ApplicationFactory
F.6.5.8 DomainManager Semantics

Add:
OMG Naming Service to Appendix F, Section F.4, Normative References
Remove:
OMG Naming Service from Main spec, Section 1.4, Normative References

Recommendation

Issue #154, 156 IDL to C++11 Type Mapping

Issue Description:
The IDL to C++11 specifications maps to native types as specified by the C++ 2011 specification, not to types as part of POSIX. Later on in the document it is said correct, it says" Each OMG IDL basic type is mapped to the listed C++ type as defined by C++11 or by the fixed-size integral types of the header <cstdint>".

Summary of the Proposal

Summary of the Proposal:

Modify (as shown below) Appendix E-3 text to clarify the IDL to C++ type mapping.

Detailed Proposal :

From:

Appendix E-3.7 , IDL TO LANGUAGE SPECIFIC MAPPINGS

The IDL to C++11 mapping uses native language types that relate to POSIX.

To:

The IDL to C++11 mapping maps basic types to C++ types as defined by C++11 or by the fixed-size integral types of the header <cstdint>. The POSIX library specification contains type definitions that relate to the C++ 11 types.

Delete:

The C, C++ and C++11 Platform Specific Model (PSM) representations are still being finalized

Summary of the Proposal (cont)

Detailed Proposal (cont) :

From:
As a consequence of these and other differences a CPP interface implementation defined in accordance with the existing mappings would not be compatible with a C++11 interface implementation from a reuse standpoint (e.g. sequence types are generated differently).
To:
There is no portability between the different OMG IDL to language mappings because of differences between the specific language mappings. An implementation using one language mapping has to be ported to another (e.g. from C++ to C++11)

From:
The transformations expressed within the tables below provide a framework for the header files that will be generated. The objective of the mapping sare to develop a programming language specific, native representations that incorporate optimizations required for efficient implementations
To:
The following tables list for each specific OMG IDL language mapping the mapping for the basic IDL data types.

Recommendation

Issue #166, Factory Release Component

Issue Description:

CFComponentManager interface is redundant. The ComponentFactory should be like the ApplicationFactory, only create and let the created object handle its destruction via the lifecycle releaseObject method. Below are the 2 methods that are part of the ComponentManager interface

1. getComponent: the Application registeredComponents should be the attribute used to obtain the application component and the DeviceManager registeredComponents should be the attribute used to obtain platform component. In 4.x there is no need to obtain a created component from the component factory.

2. releaseComponent:
Application and Platform Component must implement the releaseObject therefore calling releaseComponent on the factory is superfluous way to release a component. The ApplicationFactory does not have a releaseApplication since the releaseObject of the Application is used to terminate an application that has been created by the ApplicationFactory create member function.

Summary of the Proposal

Summary of the Proposal:

Deletion of the Component Manager interface from SCA 4.1.

Rationale: the ComponentManager is not utilized during component deployment by the CF. The componentManager provided unspecified usage to get and release components by software entities outside of the CF deployment. With the ComponentManager interface component release behavior is at 3 different levels (i.e. component lifecycle interface, Factory release object, and componentManager componentRelease) leading to confusion.

Detailed Proposal :

Delete

1. Section 3.1.3.1.1.2, ComponentManager and the subsections through section 3.1.3.1.1.2.2.2.5, Exceptions / Errors
2. Section 3.1.3.1.2.3, ComponentManagerComponent and all subsections through section Section 3.1.3.1.2.3.4 , Constraints
3. ComponentManager references in Section 2.2.4