# Comments on Software Communications Architecture Specification Version 2.2.2

**prepared by**

**SCA Working Group**

SDRF-06-A-0012-V0.00

Approved 20 April 2007

## 1 Introduction

The SDR Forum SCA Working Group has collected comments regarding Software Communications Architecture Specification Final / 15 May 2006 Version 2.2.2. These comments are respectfully submitted to the Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS) for consideration of incorporation into future versions of the SCA Specification. As a key element of its charter, the SCA Working Group is intended to supply the SDR community with a venue to evaluate and provide commentary and recommendations for change proposals against the Software Communications Architecture (SCA) Specification. This document represents the first such submission to the JTRS JPEO on behalf of the SDR Forum.

## 2 Detailed Comments

1. The *Application* interface contains four attributes, *componentNamingContexts, componentProcessIds, componentDevices,* and *componentImplemenations*, which expose the implementation details of the interface. The SCA should clearly define the use cases for these attributes or remove them from the interface.

   If use cases are defined and these attributes are kept in the SCA, the *componentNamingContexts* attribute needs amending. Section 3.1.3.2.1.4.3 of the SCA states that an *Application* will maintain a list of naming service contexts in the *componentNamingContexts* attribute. It is the understanding of the SCA Working Group that a naming context is comparable to knowing the folder or directory location, but not the object name. In order to be useful, a binding name is also required.

2. Section 3.1.3.1.4 defines the *PortSupplier* interface which contains the getPort() function. SCA Working Group recommends consideration of extending this interface which allows all provides ports to be requested in a single call. Such a function would require the definition of a PortInfo structure containing the port name and the port object reference. This function could reduce the number of calls between the CF and the WF during application instantiation and improve startup efficiency. As a further extension to the *PortSupplier* interface, the SCA Working Group recommends the addition of two new functions which allows multiple ports to be connected/disconnected with a single call. The addition of these operations would greatly reduce the number of calls between the CF and the WF during application instantiation and teardown which improves startup efficiency. The SCA Working Group understands that this addition could potentially invalidate implementations that rely on multiple getPort() invocations when connecting to a port multiple times; however, there are no requirements in the SCA today mandating that a CF make multiple getPort() invocations to retrieve the same port and this behavior should not be expected from a CF.

3. The SCA Working Group recommends clarification of the requirements for obtaining port references when the *componentSupportedInterface* (SCA Specification Section D.6.1.5.1.3) is specified. The current understanding is that a CF is not required to call getPort() on the component once found. Instead the component itself is the object to use in the connectPort() operation.
4. In the SCA Specification, the terms Resource, Device, and Service define different classes of objects (refer to SCA Specification section 2.2.3); however, definitions and interfaces are only provided for Resources and Devices. The SCA Working Group recommends that a definition and interface for Services be added to the specification.
5. The SCA Working Group recommends adding port disconnect behavior to the releaseObject() operation as an optimization for tearing down waveforms. Currently, a CF must make myriad individual disconnectPort() CORBA invocations (one for each port connection) prior to invoking releaseObject() on a *Resource*. If port disconnection requirements are added to the releaseObject() behavior of a *Resource*, then the extra CORBA invocations to disconnect the ports would be optimized to local invocations made within the *Resource* itself.
6. The current structure of the XML files only allows the CF to establish connections between the components of a single waveform, between waveform components and devices, and between waveform components and services (during application instantiation). The only way in which an application's external ports can be connected is through calls from a third-party application which uses the correct sequence of getPort() and connectPort() calls. The SCA Working Group recommends extending the SPD definition to allow an assembly to connect to other assemblies. An assemblyimplementation XML element would be added to the SPD such that a SAD could reference components that are themselves assemblies.
7. The SCA Working Group recommends that it should be possible to model sequences of an enumerated type in D.4.1.2, similar to D.4.1.1 where a simple property type can be modeled with an enumerations attribute.
8. The SCA Working Group recommends clarification of the definition of the OE. The current definition found in section 2.2.3 does not include the Board Support Package (BSP) which forms the foundation with which the O/S and middleware are built upon. It also does not mention the Log service. Lastly, it is unclear how to classify the devices and services of the platform. The definition of the OE found in section 2.2.3 does not include the devices and services although they are provided as part of a platform and not as part of a waveform. The SCA Working group further recommends clarifying Figure 3.1 of the SCA pending resolution of the OE definition. This figure should clearly distinguish between Core Framework Control/File Access on one hand and System Components (services and devices) on the other hand. Furthermore, depending on the final definition of the OE, the Systems Components should be shown outside of the OE area. Also, the SCA Working Group recommends updating Figure 3.1 to clearly depict the following CORBA services: naming service, event service and the log service.

*9.* The current requirements for the *ApplicationFactory's* create() operation mandate that component identifiers be passed as execute parameters using the format "Component_Instantiation_Identifier: Application_Name" The Application_Name field is intended to provide a "specific instance qualifier for executed components". However, the SCA places no uniqueness constraints on the Application_Name field which gets set to the create() operation's *name* parameter. Thus, it is legal for a platform to create multiple instances of an application using the same application name. When this happens, the components of the different application instances will contain identifiers that are identical. To maintain the uniqueness of identifiers, the SCA working group recommends placing uniqueness constraints on the create() operation's *name* parameter.

*10.* There is currently no specification for the *Application* interface's *identifier* attribute (inherited from the *Resource* interface). Some CFs interpret the identifier to be the *id* attribute of the *softwareassembly* element from the SAD file. However, the identifier for an application should not be the *SAD.softwareassembly.id* for two reasons. First, the *SAD.softwareassembly.id* is the identifier for the *ApplicationFactory*. If the same identifier is used for another object, then it is no longer unique. Second, if the *SAD.asoftwareassembly.id* were used for an *Application* instance, there would be no way to distinguish between multiple, simultaneous instances of the same application. The SCA Working Group recommends adding paragraph 3.1.3.2.1.4.7 to define the *identifier* attribute for an *Application*. In keeping with current *Resource* identifier requirements, the SCA Working Group further recommends using the format "Application_Identifier: Application_Name" to create a unique identifier for each *Application* instance (pending resolution of proposal 9 above). The Application_Identifier field is identical to the *id* attribute of the *sorftwareassembly* element from the SAD file. The Application_Name field is identical to the *name* parameter of the *ApplicationFactory's* create() operation.

*11.* The SCA Working Group recommends clarifying the *identifier* attribute of the *ResourceFactory* interface. Section 3.1.3.1.7.4.1 defines the attribute as the unique identifier for a *ResourceFactory* but does not specify its origin or format. The *ApplicationFactory* sets the identifier when creating the component using the format "Component_Instantiation_Identifier: Application_Name" where Component_Instantiation_Identifier is the *componentinstantiation* element *id* attribute found in the SAD file and Application_Name is the input *name* parameter of the create() operation.

12. The SCA remains ambiguous for the origin of many of the readonly attributes available in the run-time through the primary Core Framework objects. Vendors have typically imposed their own interpretations making it difficult to write portable software. We propose a table similar to the following that clarifies the interpretation of run-time parameters (note that some of the details of this table are pending the resolution of proposals 9 and 10 above):

| CF Object | Attribute | Format |
|---|---|---|
| Resource | identifier | <SAD.componentinstantiation.id>:<Application Name> |
| Device | softwareProfile | profile.filename = <SPD filename><br>profile.type = SPD (optional) |
| | label | <DCD.componentinstantiation.usagename> |
| | identifier | <DCD.componentinstantiation.id> |
| DeviceManager | deviceConfigurationProfile | profile.filename = <DCD filename><br>profile.type = DCD (optional) |
| | identifier | <DCD.deviceconfiguration.id> |
| | label | <DCD.deviceconfiguration.name> |
| | registeredServices.serviceName | <DCD.componentinstantiation.usagename> |
| ResourceFactory | identifier | <SAD.componentinstantiation.id>:<Application Name> |
| Application | identifier | <SAD.softwareassembly.id>:<Application Name> |
| | profile | profile.filename = <SAD filename><br>profile.type = SAD (optional) |
| | name | <Application Name> |
| ApplicationFactory | identifier | <SAD.softwareassembly.id> |
| | name | <SAD.softwareassembly.name> |
| | softwareProfile | profile.filename = <SAD filename><br>profile.type = SAD (optional) |
| DomainManager | identifier | <DMD.domainmanagerconfiguration.id> |
| | domainManagerProfile | profile.filename = <DMD filename><br>profile.type = DMD (optional) |