SOFTWARE COMMUNICATIONS ARCHITECTURE EXTENSIONS

FINAL / 22 December 2006

Version 2.2.2

Prepared by:

JTRS Standards
Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS)
Space and Naval Warfare Systems Center San Diego
53560 Hull Street, San Diego CA 92152-5001

TABLE OF CONTENTS

1 INTRODUCTION

The intent of this extension to the Software Communications Architecture (SCA) is to address two areas which are under specified within SCA 2.2.2. Specifically this extension addresses the deployment of non-SCA services (i.e. those other than Log, FileSystem, Event and Naming) and the introduction of a minimally intrusive mechanism to manage and optimize application deployment. The underlying design constraint behind this extension was to introduce an approach that was fully backward compatible with the existing 2.2.2 specification. There are behavioral requirements that must be implemented within an SCA 2.2.2 compliant Core Framework if the capabilities described within this extension are supported; however the decision of whether or not to include these capabilities is at the discretion of the Platform provider.

2 SCA MAIN DOCUMENT EXTENSIONS

2.1 REGISTERSERVICE BEHAVIOR (3.1.3.2.3.6.7.3)

The *registerService* operation shall, upon successful service registration of a non-SCA service with an input name parameter in the “identifier\type” format, make the value provided in the “identifier” potion of the name accessible via the *domainfinder servicename* mechanism.

The *registerService* operation shall, upon successful service registration of a non-SCA service with an input name parameter in the “identifier\type” format, make the value provided in the “type” potion of the name accessible via the *domainfinder servicetype* mechanism.

2.2 UNREGISTERSERVICE BEHAVIOR (3.1.3.2.3.6.8.3)

The *unregisterService* operation shall remove non-SCA services (i.e. those with a name in the “identifier\type” format) by matching either a fully qualified name in the “identifier\type” format or a simple name with only the “identifier” portion.

2.3 CREATE BEHAVIOR (3.1.3.2.2.5.1.3)

The *create* operation shall recognize application deployment channel preferences contained within an Application Deployment Descriptor file if the CF implementation provides enhanced deployment support via the use of both a Deployment Platform Descriptor and an Application Deployment Descriptor file.

The *create* operation shall recognize a property which is a CF *Properties* type with an id of “DEPLOYMENT_CHANNEL” and a value that is a string sequence if the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file.

The *create* operation shall recognize channel preferences contained within a “DEPLOYMENT_CHANNEL” property contained within the initConfiguration parameter if the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file.

The *create* operation shall attempt to allocate an application to the Deployment Platform Descriptor file channel alternatives provided within a “DEPLOYMENT_CHANNEL” property or an Application Deployment Descriptor file in a sequential manner.

The *create* operation shall utilize channel preferences expressed within a “DEPLOYMENT_CHANNEL” property rather than those contained within an Application Deployment Descriptor file if both exist and the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file.

The *create* operation shall recognize a deployment option with a deployedname attribute value of “DEFAULT” which matches all application instance names that are not explicitly identified by a deployedname attribute value within the same descriptor file if the CF implementation provides enhanced deployment support via the use of an Application Deployment Descriptor file.

For connections to a non-SCA service using the servicename type of the *domainfinder* element, the *create* operation will search for a matching name from the set of service name identifiers that have been registered with the domain. For connections to a non-SCA service using the servicetype type of the *domainfinder* element, the *create* operation will search for a matching type from the set of service types that have been registered with the domain. The search strategy used to select a specific instance of a service type when multiple instances of the same service type have been registered with the domain is implementation dependent.

For *domainfinder* element “servicetype” connections to a non-SCA service whose service type is provided by a service contained within a *channel* element servicelist, the *create* operation shall only attempt to establish connections to services within the list if the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file. If multiple instances of the same service type exist with the servicelist, then an implementation dependent search strategy used to select a specific instance.

2.4 CREATE EXCEPTIONS/ERRORS (3.1.3.2.2.5.1.5)

The *create* operation shall raise the InvalidInitConfiguration exception when the input initConfiguration parameter “DEPLOYMENT_CHANNEL” property contains an invalid channel reference. The InvalidInitConfiguration invalidProperties parameter shall identify the invalid channels.

The *create* operation shall raise the CreateApplicationError exception when the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file but the CF is not able to allocate the application to any of the provided channel alternatives.

The *create* operation shall raise the CreateApplicationError exception when the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor file and a *domainfinder* element “servicetype” connection to a non-SCA service whose service type is provided by a service contained within a *channel* element servicelist can not be established to a service identified within that list.

2.5 DEVICE MANAGER GENERAL BEHAVIOR (3.1.3.2.4.5)

If a non-SCA service is deployed by the device manager, the device manager shall supply execute operation parameters consisting of:

1. Device manager IOR – The ID is “DEVICE_MGR_IOR” and the value is a string that is the *DeviceManager* stringified IOR.
2. Service Name – The ID is “SERVICE_NAME” and the value is a string in an “identifier\type” format where the identifier corresponds to the DCD *componentinstantiation usagename* element and the type corresponds to a service type repository identifier from the SCD.
3. The execute (“execparam”) properties as specified in the DCD for a *componentinstantiation* element. The device manager shall pass the componentinstantiation element “execparam” properties that have values as parameters. The device manager shall pass “execparam” parameters’ IDs and values as string values.

2.6 SOFTWARE COMPONENT DESCRIPTOR (3.1.3.5.2)

A Software Component Descriptor (SCD) contains information about a specific SCA software component (*Resource, ResourceFactory, Device* or non-SCA service).

2.7 GENERAL APPLICATION REQUIREMENTS (3.2.1)

An application’s dependencies to the log, file manager, file system, CORBA Event Service, CORBA Naming Service, and non-SCA services are specified as connections in the SAD using the *domainfinder* element

2.8 DEPLOYMENT PLATFORM DESCRIPTOR

A Deployment Platform Descriptor (PDD) identifies the logical relationships between platform resources within the operating environment’s registered services and devices. A Deployment Platform Descriptor file shall have a “.pdd.xml” extension. The use of the PDD is optional within a system, however if it is used the reference to this file will be made from the DMD file. A Deployment Platform Descriptor File may be used to exert a greater degree of control over the application deployment process. The file contains information that describes the composition (i.e. included services and devices) of virtual channels within a platform domain.

2.9 APPLICATION DEPLOYMENT DESCRIPTOR

An Application Deployment Descriptor (ADD) contains precedence lists that are used for deploying application instances within a platform domain. An Application Deployment Descriptor file shall have an “.add.xml” extension. The use of the ADD is optional within a system, however if it is used the reference to this file will be made from a SAD file. An Application Deployment Descriptor file contains application names and references the virtual channels defined in the Deployment Platform Descriptor file.

3 SCA APPENDIX A EXTENSIONS

None

4 SCA APPENDIX B EXTENSIONS

None

5 SCA APPENDIX C EXTENSIONS

None

6 SCA APPENDIX D EXTENSIONS

6.1 DOMAIN PROFILE

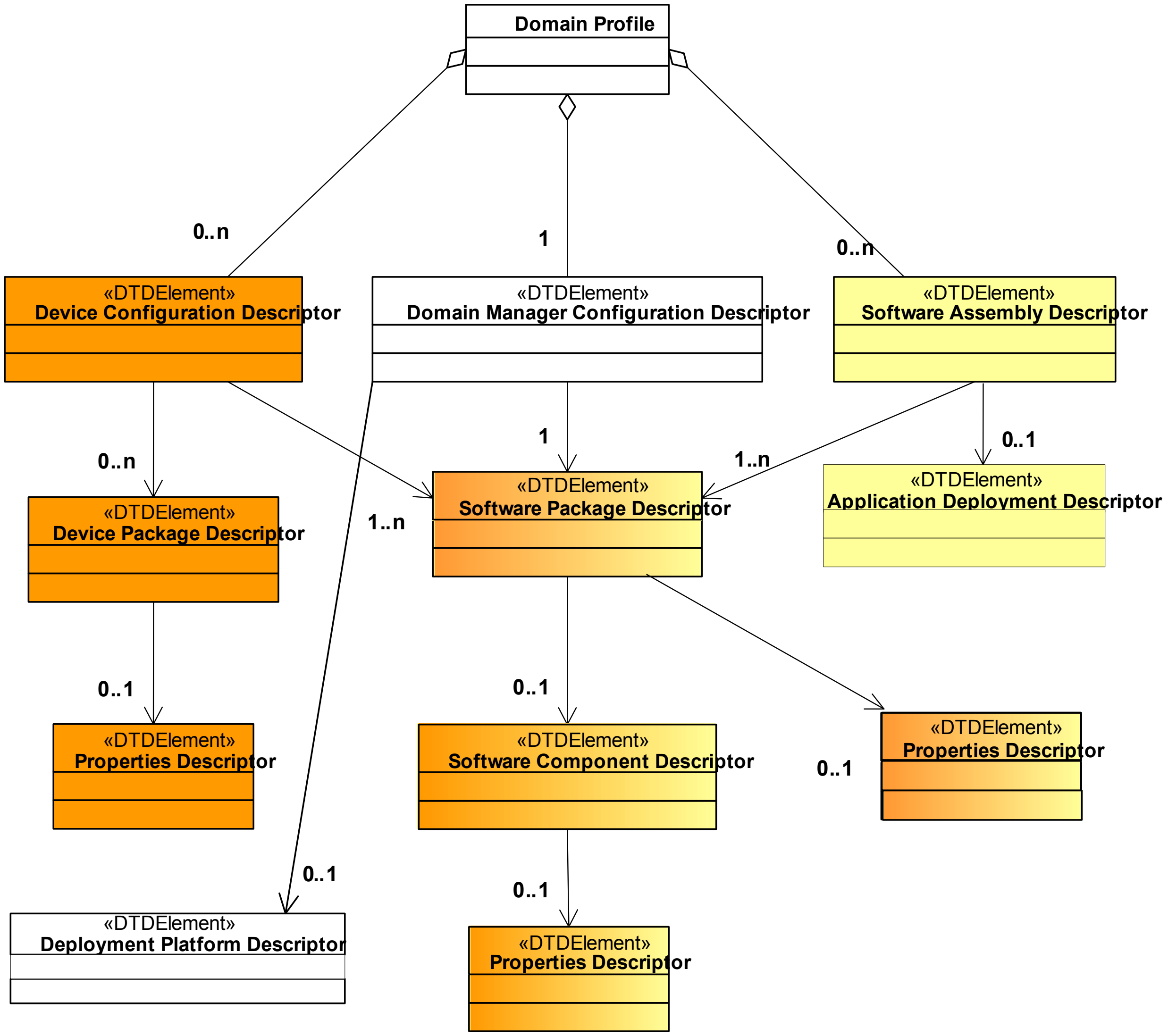

Figure 6-1: Relationship of Domain Profile XML File Types

6.2 SOFTWARE COMPONENT DESCRIPTOR (D.5)

This descriptor file is based on the CORBA Component Descriptor specification. The SCA components CF *Resource*, CF *Device*, and CF *ResourceFactory* and the non-SCA service components that are described by the software component descriptor are based on the SCA CF specification, and the following specification concentrates on definition of the elements necessary for describing the ports and interfaces of these components

6.3 SCD COMPONENTREPID (D.5.1.2)

The *componentrepid* uniquely identifies the interface that the component is implementing. The *componentrepid* may be referred to by the *componentfeatures* element. The *componentrepid* is either derived from the CF *Resource*, CF *Device*, or CF *ResourceFactory* or represents a non-SCA service. For non-SCA services the repid will be used as the type identity for *domainfinder* servicetype searches.

6.4 SCD COMPONENTTYPE (D.5.1.3)

The *componenttype* describes properties of the component. For SCA components, the component types include resource, device, resourcefactory, domainmanager, log, filesystem, filemanager, devicemanager, namingservice, eventservice and service. The “service” type is used for all non-SCA services.

6.5 SCD COMPONENTFEATURES (D.5.1.4)

The *componentfeatures* element does not need to contain information when used in conjunction with a non-SCA service.

6.6 SCD INTERFACES (D.5.1.5)

For non-SCA services the *interfaces* element is made up of zero to many *interface* elements.

```
<!ELEMENT interfaces
     ( interface* )>
```

For non-SCA services the *interface* element describes any services interfaces in addition to the one identified in the *componentrepid* element that need to be registered as services. The name attribute value contains the unique “identifier” portion of the “identifier\type” format service name. The repid attribute is the unique repository id of the interface, which contains the “type” portion of the name. This information should be passed to a service as execparams and the service will need to register these items as services using the “identifier\type” format. For non-SCA services the inheritsinterface element is not expected to contain a value.

```
<!ELEMENT interface
     ( inheritsinterface*) >
<!ATTLIST interface
     repid     CDATA     #REQUIRED
     name      CDATA     #REQUIRED>
```

```
<!ELEMENT inheritsinterface EMPTY>
     <!ATTLIST inheritsinterface
    repid CDATA     #REQUIRED
```

6.7 SCD PROPERTYFILE (D.5.1.6)

No *propertyfile* element entries are expected for non-SCA services.

6.8 SAD SOFTWAREASSEMBLY (D.6.1)

```
<!ELEMENT softwareassembly
     ( description?
     , componentfiles
     , partitioning
     , assemblycontroller
     , connections?
     , externalports?
     , deploymentprefs?
     )>
<!ATTLIST softwareassembly
     id         ID         #REQUIRED
     name CDATA     #IMPLIED
     version    CDATA      #IMPLIED>
```

6.8.1 deploymentprefs

The optional *deploymentprefs* element is a reference to a local file. See section D.2.1.4.1 for the definition of the *localfile* element. The file refers to an Application Deployment Descriptor file.

```
<!ELEMENT deploymentprefs
     ( localfile
     )>
```

6.9 SAD FINDBY (D.6.1.5.1.1.3)

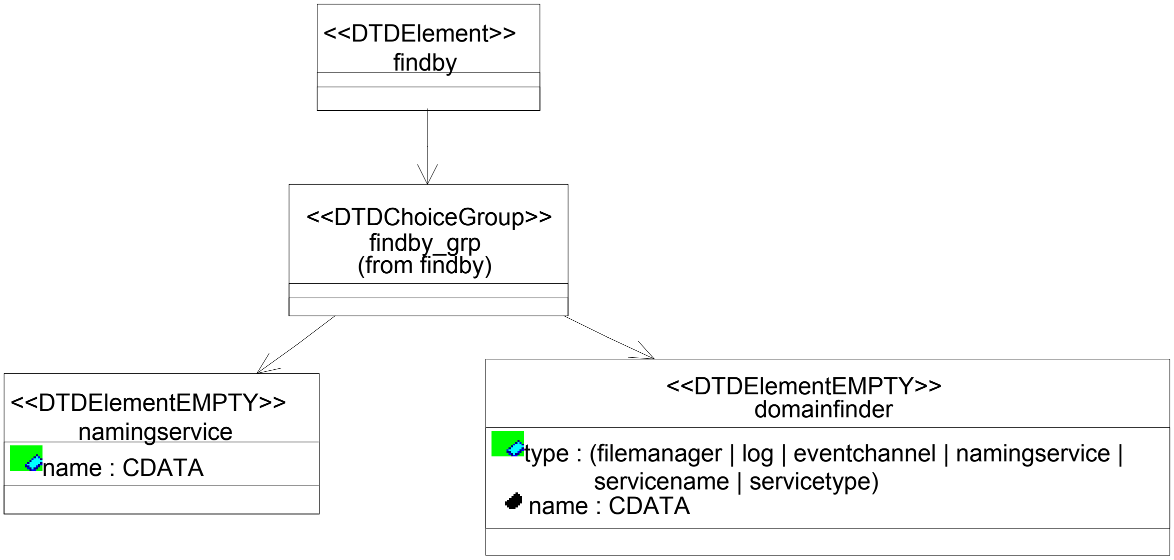

Figure 6-2. ***findby*** **Element Relationships**

6.10 SAD DOMAINFINDER (D.6.1.5.1.1.5)

The *domainfinder* element is a child element of the *findby* element. The *domainfinder* element is used to indicate to the CF *ApplicationFactory* the necessary information to find an object reference that is of specific type and may also be known by an optional name within the domain. The valid type attributes are “filemanager”, “log”, “eventchannel”, “namingservice”, “servicename” and “servicetype”. If a name attribute is not supplied, then the component reference returned is the CF *DomainManager's FileManager*, or Naming Service corresponding to the type attribute provided. If a name attribute is not supplied and the type attribute has a value of “log”, then a null reference is returned. The type attribute value of “eventchannel” is used to specify the event channel to be used in the OE’s CORBA Event Service for producing or consuming events. If the name attribute is not supplied and the type attribute has a value of “eventchannel” then the Incoming domain management event channel is used. The type attribute value of “servicename” is used to locate registered non-SCA services on a per name basis. The type attribute value of “servicetype” is used to locate registered non-SCA services on a per type basis where the corresponding type information is provided in a service Software Component Descriptor file.

```
<!ELEMENT domainfinder EMPTY>
<!ATTLIST domainfinder
     type (filemanager | log | eventchannel | namingservice |
          servicename | servicetype) #REQUIRED
     name CDATA     #IMPLIED>
```

6.11 DEPLOYMENT PLATFORM DESCRIPTOR

This section describes the XML elements of the Deployment Platform Descriptor (PDD) XML file; the *deploymentplatform* element. The intent of the PDD is to provide a means of describing the collection of services and devices that are associated with a virtual channel. The knowledge of the channel composition can be utilized as part of an overall systems engineering strategy to control the allocation of applications to system resources. Another use of the information could be to improve the efficiency of application deployment as the channel elements would be used to constrain the search space for the allocation of individual application components. The use of the PDD is optional within a system, a system designer is free to use allocation properties or other approaches to manage the allocation of application components to platform resources.

6.11.1 *deploymentplatform*

The *deploymentplatform* element contains the layout of the virtual channels within a platform domain.

```
<!ELEMENT deploymentplatform
     ( description?
     , platformlayout
     ) >
```

6.11.2 description

The optional *description* element of the PDD may be used to provide information about the platform domain.

```
<!ELEMENT description (#PCDATA)>
```

6.11.3 platformlayout

The *platformlayout* element references the definitions of the virtual channels that exist within the platform domain.

```
<!ELEMENT platformlayout
     ( channel+
     )>
```

6.11.4 channel

The *channel* element in the PDD defines the collections of devices and services that are used by the CF *ApplicationFactory* as target resource pools for application deployment. The *channel* element's name attribute contains the identifier for the channel that is used by the CF ApplicationFactory and the Application Deployment Descriptor.

```
<!ELEMENT channel
     ( devicelist?
     , servicelist?
     ) >
<!ATTLIST channel
     name ID   #REQUIRED>
```

6.11.5 devicelist

The optional *devicelist* element in the PDD defines the collection of devices for a given channel that are used by the CF *ApplicationFactory* as target resource pools for application deployment.

```
<!ELEMENT devicelist
     ( deviceref*
     )>
```

6.11.6 deviceref

The *deviceref* element is used to reference a *componentinstantiation* element which is part of the channel. The refid attribute points to a *componentinstantiation* identifier for a device that has registered with the platform.

```
<!ELEMENT deviceref EMPTY>
<!ATTLIST deviceref
     refid          CDATA          #REQUIRED>
```

6.11.7 servicelist

The optional *servicelist* element in the PDD defines the collection of services for a given channel that are used by the CF *ApplicationFactory* as target resource pools for application deployment.

```
<!ELEMENT servicelist
     ( serviceref*
     )>
```

6.11.8 serviceref

The *serviceref* element identifies a service which is part of the channel. The servicename attribute is identical to a *usagename* identifier for a service that has registered with the platform.

```
<!ELEMENT serviceref     EMPTY>
<!ATTLIST serviceref
     servicename        CDATA          #REQUIRED>
```

6.12 APPLICATION DEPLOYMENT DESCRIPTOR

This section describes the XML elements of the Application Deployment Descriptor (ADD) XML file; the *deploymentprecedence* element. The intent of the ADD is to provide prioritized lists of deployment alternatives for application instances.

6.12.1 *deploymentprecedence*

The *deploymentprecedence* element contains the relationship between application instances and their candidate virtual channels.

```
<!ELEMENT deploymentprecedence
     ( description?
     , deploymentoptions
     ) >
```

6.12.2 description

The optional *description* element of the ADD may be used to provide information about the application.

```
<!ELEMENT description (#PCDATA)>
```

6.12.3 deploymentoptions

The *deploymentoptions* element refers to the definition of the deployment preferences that exist for each application instance.

```
<!ELEMENT deploymentoptions
     ( deploymentoption+
     )>
```

6.12.4 deploymentoption

The *deploymentoption* element in the ADD identifies the ordered list of channels that provide deployment alternatives for a specific application instance. The deployedname attribute corresponds to a named application instance (e.g. the name parameter passed to the CF *ApplicationFactory create* operation). The optional *description* element may be used to provide information about the application instance.

```
<!ELEMENT deploymentoption
     ( description?
     , channelref+
     )>
<!ATTLIST deploymentoption
     deployedname   CDATA     #REQUIRED>
<!ELEMENT description (#PCDATA)>
```

6.12.5 channelref

The *channelref* element is used to reference a *channel* element from the Deployment Platform Descriptor which provides a deployment alternative. The refname attribute points to a *channel* element name attribute that identifies a channel.

```
<!ELEMENT channelref EMPTY>
<!ATTLIST channelref
     refname    CDATA     #REQUIRED>
```

6.13 DMD DOMAINMANAGERCONFIGURATION (D.8.1)

```
<!ELEMENT domainmanagerconfiguration
      ( description?
      , domainmanagersoftpkg
      , deploymentlayout?
      , services
      )>
<!ATTLIST domainmanagerconfiguration
     id         ID   #REQUIRED
     name CDATA      #REQUIRED>
```

6.13.1 deploymentlayout

The optional *deploymentlayout* element is a reference to a local file. See section D.2.1.4.1 for the definition of the *localfile* element. The file refers to a Deployment Platform Descriptor file.

```
<!ELEMENT deploymentlayout
     ( localfile
     )>
```