Joint Program Executive Office
Joint Tactical Radio System

SCA Service and Multi Channel Deployment Extensions

JTRS Interface Control Working Group (ICWG)

08 DEC 2006

Solution Approach

- **These options represent a minimally intrusive approach to introducing more flexibility into the SCA Service deployment mechanism**
  - It aims to maintain backward compatibility with existing SCA versions
  - It depends on service connectivity being established by name
- **What topics are not addressed by this proposal**
  - Whether or not SCA services and devices should be merged into one data type
  - If a service interface needs to be introduced into the SCA
  - Whether or not Services can call other services (2 way service connectivity)
  - Overall SCA service characteristics
  - How commercially acquired versus internally developed Services need to be differentiated
  - How are services integrated into the Capacity Model

Background - Existing SCA Services

- Non-hardware (software-only) resources provided by the system for use by applications are generically referred to as “services”,
  - **SCA does not specify an explicit interface that must be realized by a service component**
- Register Service (domain and device manager) operation accepts a CORBA object as a Service to be introduced into the domain.
  - **Allows the freedom for any type component or object to be represented as a service**
- Constraints introduced within the domain profile do not permit arbitrary user defined services to be deployed and connected (The SCA currently identifies the following Services)
  - Log
  - Naming
  - Event
  - File (although named File Service interfaces these should not be thought of as SCA “services”)

SCA Descriptor Service Deployment Extensions

Service Support - Componenttype

- **The SCD *componenttype* describes properties of the component. For SCA components, the component types include resource, device, resourcefactory, domainmanager, log, filesystem, filemanager, devicemanager, namingservice and eventservice.**

<!ELEMENT componenttype (#PCDATA)>

Necessary Enhancements

1. The componenttype text will be modified to include an additional type "service". The "service" type will be used as the identifier for non-application components that realize service interfaces other than those identified in the SCA service section.
2. The componentfeatures element will be made optional for the SCD file, it will be included if the component represents an SCA defined element, but will not be included if the componenttype is a service.
3. The componentrepid wording will be changed to allow for the declaration of items that are not derived from the resource or resourcefactory interfaces (i.e. the service interface repository identifier will be entered). The service interface contained within the componentrepid element will identify the type identity of the service which is used as for *servicetype* searches.

Service Support - Domainfinder

- **The *domainfinder* element indicates that the necessary information to find an object reference that is of specific type and an optional name is located within the domain. The valid type attributes are “filemanager”, “log”, “eventchannel”, and “namingservice”.**

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

Necessary Enhancements

1. The type enumeration should be extended to include two items : “servicename” and “servicetype”, this maintains backwards compatibility with existing CF implementations as well as the special semantics associated with the filemanager and namingservice services
   - The new types will be used to locate registered services that realize interfaces other than those identified as SCA services. Services that exist within the SCA main text will be located using distinct types within the domainfinder list of types
   - This approach provides an extensible mechanism for establishing connectivity to additional services

Service Support - Componentinstantiation

- **The DCD *componentinstantiation* element describes a particular instantiation of a component. The *componentinstantiation*'s id attribute uniquely identifies the component. The *componentinstantiation* contains a *usagename* element that is intended for an applicable name for the component. For a component service type (e.g., Log), the *usagename* element is not optional and needs to be unique for each service type.**

Necessary Enhancements

1. No changes, the DCD/Device Manager is conventionally responsible for service deployment, a scenario exists for platform services deployed outside of a DCD, but that is not the primary approach and is not explicitly addressed (although it should work with the proposed mechanism if a properly structured service name exists)
2. No changes, the usagename is the unique identity element for a service name. All user identified services map to the type "service" within componenttype, so uniqueness is across all of the services registered within the domain. Usagename provides the portion of the registered service name that is matched in establishing *servicename* connections.

Service Support – DMD elements

- **The *domainmanagersoftpkg* element refers to the SPD for the *DomainManager*. This SPD describes the *DomainManager* implementation and refers to the descriptor that specifies the *usesports* for services used by the *DomainManager*.**
- **The *services* element determines which service instances the DomainManager uses**

<!ELEMENT services
(service+) >

Necessary Enhancements

1. The SCD referred to by the domainmanagersoftpkg reference will not require any structural changes. The uses port repid refers to the repository identifier of the interface supported by the service. The usesname corresponds to the name provided in the service SCD interface definition.
2. The service element will not require any structural changes, the usesidentifier contains the name provided in the service SCD interface definition, existing SCA services will be located using the same domainfinder mechanism. New services will be found via the domainfinder as well; however the type will be a "*service name* or *type*" and locating the element within the domain will be dependent on the name or type identifier assigned to the service instance.

SCA Service Deployment CF Behavior Extensions

Service Support – Device Manager

- **When a device manager deploys a service it supplies execute parameters for the Device manager IOR and the service name. These two parameters facilitate the registration of the service within the domain.**
- **The registerservice operation adds the new service into the device manager registeredservices attribute, which is an object reference and name sequence of services that have registered.**
- **deviceManager registerservice registers the service with the domain. This operation adds the name of the new service into the domain manager's knowledge base**

Necessary Enhancements

1. When a device manager deploys a user defined service it is expected that all CORBA enabled services will have a corresponding SPD and SCD, the devicemanager will need to follow the path provided by the service's componentfile reference in order to obtain the service repository id (service type) from its descriptor
2. When a devicemanager launches a service it will provide the service name to the service implementation as a single string in a "name/type" format (where name is the provided component usagename and type is the component interface) – this change will also require a rewording of the device manager service launching section.
3. A new requirement will be introduced that requires any new service that registers with the domain using the naming convention from (2) will subsequently be accessible via the domainfinder mechanism. The left side of the "/" will be matched for servicename requests and the right side for servicetype(s)

Service Support - Applications

- **An application's dependencies to the log, file manager, file system, CORBA Event Service, and CORBA Naming Service will be specified as connections in the SAD using the *domainfinder* element.**

Necessary Enhancements

1. The application text needs to be extended to mention that platform defined services can also be accessed via the domainfinder element.
2. No requirement (shall statement) exists in the document to enforce this behavior. It is not recommended that a requirement be added however we should provide guidance (outside of the SCA document) that describes how to use the capability.

Service Support – Application Factory

- **The *create* operation establishes application connections that are specified in the SAD *domainfinder* element. The *create* operation obtains object references to the required *Port* interfaces in via *PortSupplier::getPort* operation.**

Necessary Enhancements

1. No changes are required in the mechanism to build a connection to a platform service, the uses side of the connection is obtained via the getPort mechanism, the particular service instance (provides side) is obtained by requesting a specific element from the domainfinder.

Service Connection Establishment – Application Factory

Necessary Enhancements

- When *domainfinder – servicename* connections are requested, the Domain Management Functions will search through collection of service names registered with the domain for the specific name
- When *domainfinder – servicetype* connections are requested, the Domain Management Functions will search the collection of registered services for a service instantiation that corresponds to the specific type. The search strategy used to select an instance when multiple instances of the same type are registered is implementation dependent.

Extensions for Optimized Deployment in a Multi-Channel Environment

Deployment Support for Multi-Channel Environments

Introduce two new descriptor files which identify the platform composition and waveform channel precedence order

- This alternative does not require a modification of the ApplicationFactory create operation interface or any other CF operation interfaces or parameters.
- The new XML files are optional (inclusion is dictated by the platform), hence it does not impact existing CF implementations or existing fielded radios.
- The proposal impacts the behavior of the ApplicationFactory (Domain Management functions).
- The ApplicationFactory create operation deviceAssignments parameter, when specified, can still be used to further constrain deployment (will continue to be used as today).
- A platform integrator can opt to use the existing allocation property mechanism to constrain deployment in lieu of this option

Proposed Deployment Platform File

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT deploymentplatform
      ( description?
      , platformlayout
      ) >
<!ELEMENT description (#PCDATA)>
<!ELEMENT platformlayout
      ( channel+
      )>
<!ELEMENT channel
      ( devicelist?
      , servicelist?
      ) >
<!ATTLIST channel
      name    ID          #REQUIRED>
<!ELEMENT devicelist
      ( deviceref*
      )>
<!ELEMENT deviceref     EMPTY>
<!ATTLIST deviceref
      refid             CDATA                 #REQUIRED>
<!ELEMENT servicelist
      ( serviceref*
      )>
<!ELEMENT serviceref    EMPTY>
<!ATTLIST serviceref
      servicename       CDATA       #REQUIRED>
```

The channel name corresponds to a virtual channel name within the platform and not necessarily the physical channel.

Example Use of Deployment Platform File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deploymentprecedence SYSTEM “deploymentplatform.dtd">
<deploymentplatform>
    <platformlayout>
        <channel name=“Channel1”>
            <devicelist>
                <deviceref refid=“DCE:aaa”/>
                <deviceref refid=“DCE:bbb”/>
                <deviceref refid=“DCE:ccc”/>
                <deviceref refid=“DCE:ddd”/>
                <deviceref refid=“DCE:eee”/>
            </devicelist>
            <servicelist>
                <serviceref servicename=“MyTimer1”/>
            </servicelist>
        </channel>
        <channel name=“Channel2”/>
        <channel name=“Channel3”/>
        <channel name=“Channel4”/>
    </platformlayout>
</deploymentplatform>
```

DMD Changes

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

Proposed Deployment Preferences File

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT deploymentprecedence
    ( description?
    , deploymentoptions
    ) >
<!ELEMENT deploymentoptions
    ( deploymentoption+
    )>
<!ELEMENT deploymentoption
    ( description?
    , channelref+
    )>
<!ATTLIST deploymentoption
    deployedname      CDATA #REQUIRED>
<!ELEMENT description (#PCDATA)>
<!ELEMENT channelref EMPTY>
<!ATTLIST channelref
    refname      CDATA#REQUIRED>
```

The deploymentoption deployedname corresponds to a provided application instance. A value of DEFAULT provides the default preferences for the assembly.

The channelref refname corresponds to a channel name defined in the platformlayout element of the deploymentlayout descriptor.

Example Use of Deployment Preferences File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deploymentprecedence SYSTEM “deploymentprecedence.dtd">
<deploymentprecedence>
    <deploymentoptions>
        <deploymentoption deployedname=“SINCGARS”/>
            <description>Platform G Channel Alternatives for SINCGARS
    instantiation</description>
            <channelref refname="Channel1”/>
            <channelref refname="Channel2”/>
            <channelref refname="Channel4”/>
        </deploymentoption>
         <deploymentoption deployedname=“DEFAULT”/>
            <description>Platform G Default Application Channel
    Precedence</description>
            <channelref refname="Channel2”/>
            <channelref refname="Channel3”/>
         </deploymentoption>
    </deploymentoptions>
</deploymentprecedence>
```

SAD Changes

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

<!ELEMENT deploymentprefs
(localfile)>
```

Proposal Implications

- **The deployment platform information will be provided to the platform via the DMD**
  - If a platform wishes to use this capability they must create and process an additional descriptor file – and the DMD structure will have to be modified
  - The location of the deployment file does not need to be exposed on an interface through a "profile like" attribute however the information contained within the file needs to be accessible by the Domain Management interface components [Unless new services or devices are added to the platform then this file should not have to be changed. If necessary it is the hope that some type of installation service would be responsible for dynamically updating this information]
- **The deployment preferences information will be provided to the platform via the SAD**
  - If a platform wishes to use this capability they must create and process an additional descriptor file – and the SAD structure will have to be modified. Although this file is linked to the SAD it is the intent that a System Integrator or a collaboration between the integrator and developer will have the responsibility of developing the file, the file pointer design of this approach is intended to be minimally intrusive.
- **The Domain Management operations will need to be configured to recognize the "DEPLOYMENT_CHANNEL" property**
- **In addition to the channel references specified in the deployment options element, a preferred channel can be specified through the inclusion of a property with an ID="DEPLOYMENT_CHANNEL" and a string sequence value equal to an ordered list of valid platform channel configurations as part of the create operation initConfiguration properties. A channel specified here will take precedence over any specified in the channel precedence list. An invalid channel referenced within the channel precedence list will result in the invalidinitconfiguration exception being thrown.**

Proposal Implications

- **The Domain Management operations will need to recognize the deployment option with the deployedname of "DEFAULT" as the channel preference for all unspecified Application component instantiations**
- **If a preference file exists, Domain Management will have to maintain knowledge of the deployment information (i.e. the application to channel preferences and the contents of the channels)**
  - As DeviceManagers, Devices and Services register into the system Domain Management will update the inventory (implementation dependent) vis-à-vis the platform layout
- **When an application with a stated deployment preference is launched, the application creation operation will search its constrained device preference list (sequentially) to see if an identified channel configuration can support deployment of the application, if none of the specified configurations can accommodate the application, then the creation will fail**
- **When an application with a stated deployment preference is launched, the application creation method will search its constrained list of services for "*servicetype*" connections. If the type exists in the list, but none of the services have been registered or support the connection request, then the creation will fail. Multiple occurrences are handle in an implementation dependent manner.**