SCA 4.1 Domain Component Type Uniformity

Document WINNF-15-R-0013

Version V1.0.0

23 April 2015

Driving the future of radio communications and systems worldwide

Terms and Conditions

Intellectual Property Rights

Use this chart in all contributions

Proposal

This document contains a proposal to change the Draft SCAv4.1 specification for the domain management interfaces to utilize the ComponentType.

Proposal author:

- Hugues Latour, Communication Research Centre Canada (CRC)
- Jerry Bickle, Raytheon

Proposal reviewers:

- Chuck Linn, Harris
- Francois Levesque, NordiaSoft
- Kevin Richardson, MITRE
- Sarah Miller, Rockwell Collins

Recommendation

SCA v4.1 Domain Component Data

Topics

Description of the Issue

Summary of the Proposal

Detailed Proposal

Specifications Changes (found in a word document)

- Main Specification Changes
- IDL Specification Changes

Description of the Issue

The SCA 4 architecture introduce a push approach, which eliminated the need to pull or call back to get component information, which is not the case for DomainManagementObjectAddedEventType, ApplicationFactory::create return type, and DomainManager applications attribute.

Also the information pushed out either by registration, creation, or event should be the same information that is pulled but this is not the case for:

1. DomainManagementObjectAddedEventType
2. ApplicaitonFactory::create return type.
3. DomainManager::installApplication return type.
4. DomainManager applications and application factories attributes

To be consistent with architecture, ComponentType should be used in these areas as is in registration.

Rationale for Change

1. Domain Manager Clients are already impacted from SCA 4 interface changes
   - SCA 4 Domain Manager Interfaces changes impact
     - Domain Manager Device Managers, Applications, and Application Factories attributes types
     - ApplicationFactory create operation return type
     - InstallApplication operation return type
2. Since client Impact already exist at the Domain Manager interface level then it makes sense for SCA 4.1 to achieve a uniform and consistent approach on the usage of ComponentType
3. Applications and Application Factories recommendation approach is similar to SCA 4 Domain Manager's managers attributes change.
4. Changing Domain added and removed object events for usage of ComponentType also makes sense since client code using existing event types will have to change since the DomainManager interface changed on how one pulls the information.
   - The added benefit is no pull is required to get Device Manager and Application information.
5. Remove profile from interfaces since in ComponentType and better consistency since not all interfaces have profile.
6. Retain Identifier interface and identifier/name interface attributes to be able retrieve from a Component.

Summary of the Proposal

1. For Application Installation return back a Component Type
2. For Application instantiation return back a ComponentType
3. Replace DomainManagementObjectAddedEventType and DomainManagementObjectRemovedEventType with ComponentChangeEventType
4. For DomainManager applications and application factories attributes return Components type.
5. 3.1.3.1.3.17 ComponentType description gives details on fields so this information is not repeated for returning or registering component, or giving out ComponentType information.
6. State who has responsibility for ComponentType's fields.
7. Remove profile from interfaces. Removed Profile Name parameter for Device Component deployment since profile attribute was removed.
8. Retain Identifier interface and identifier/name interface attributes

Detailed Proposal

3.1.2.2.3.6 ComponentChangeEventType

```
enum ComponentChangeType
        {
        ADDED,
        REMOVED,
        };


struct ComponentChangeEventType
        {
        string producerId;
ComponentChangeType componentChange;
CF::ComponentType domainComponent
        };
```

Remove 3.1.2.2.3.4 SourceCategoryType

Remove 3.1.2.2.3.5 DomainManagementObjectRemovedEventType

Remove 3.1.2.2.3.6 DomainManagementObjectAddedEventType

Update Appendix C and IDL file.

Detailed Proposal

For the Remove Component Change Event either add the following statement

“The ComponentType’s componentObject field should be nil reference and providesPorts field should be a zero size sequence length.” at ComponentChangeEventType description or at each component semantics for the removed component change event.

Profile – null string

specializedInfo – zero size

Detailed Proposal

1. Remove ApplicationType and ApplicationFactoryType

3.1.3.1.3.22 ApplicationType

3.1.3.1.3.23 ApplicationFactoryType

2. Update Appendix C and Remove ApplicationTypes IDL file.

Detailed Proposal

3.1.3.1.3.17 ComponentType

Modify Statement

The ComponentType structure defines the basic elements of a component.

The identifier field is the id of the component as specified through its createComponenti~~on~~/execute parameters.

The profile field is either the component's profile filename or the profile itself (the DomainManagerComponent uses a DMD, the ApplicationFactoryComponent and ApplicationManagerComponent use a SAD, the DeviceManagerComponent uses a DCD and all of the other components use an SPD). The application's SAD filename is an absolute pathname relative to a mounted FileSystemComponent and the file is obtained via the DomainManagerComponent's FileManagerComponent. ApplicationComponent Files referenced within the profile are obtained via a DomainManagerComponent's FileManagerComponent. The DCD and BasePlatformComponent filename is an absolute pathname relative to a mounted FileSystemComponent and the file is obtained via a DeviceManagerComponent's FileSystemComponent.

The type field is the value of the SCD componenttype element described in Appendix D-1 that maps to ComponentEnumType.

The componentObject field is the object reference of the component.

The providesPorts field is a sequence containing the ~~static~~ ports provided by the component.

The SpecializedInfo is a variant field that is used by to provide supplemental, component specific information.

Detailed Proposal

3.1.3.1.1.1.5.1.4 ComponentFactory Returns

The *createComponent* operation shall return a ComponentType ~~structure that contains a reference to the created component~~. The returned ComponentType ‘s identifier, componentObject, profile, and type fields are as specified in 3.1.3.1.3.17 ComponentType. The returned ComponentType may set its providesPorts field.

Detailed Proposal

3.1.3.1.2.1.4 BaseComponent Constraints

Move statement to Semantics

"A BaseComponent **may** register its component reference via the ComponentRegistry interface. The registering BaseComponent sets its ComponentType's identifier, componentObject, and type fields as specified in 3.1.3.1.3.17 ComponentType. A registering BaseComponent may set its ComponentType's providesPorts field."

Detailed Proposal

3.1.3.3.1.1 ApplicationManager – Update Section

3.1.3.3.1.1.2 UML

Update Figure 3-19

Remove Section 3.1.3.3.1.1.4.1 profile

3.1.3.3.1.1.5.2 getProvidesPorts

3.1.3.3.1.1.5.2.1 Brief Rationale

The *getProvidesPorts* operation is used to retrieve the application external provides ports as defined in the associated SAD (ApplicationManagerComponent's profile ~~attribute~~). This operation overrides the definition in *PortAccessor::getProvidesPorts* section 3.1.3.2.1.2.5.3.

3.1.3.3.1.1.5.2.3 Behavior

The *getProvidesPorts* operation returns the external provides port object references for the external provides ports as stated in the associated SAD (ApplicationManagerComponent's profile ~~attribute~~).

3.1.3.3.1.1.5.2.4 Returns

SCA53 The *getProvidesPorts* operation shall return the object references that are associated with the input provides port names for the application external ports as identified in the associated SAD (ApplicationManagerComponent's profile ~~attribute~~).

Update Appendix C and CFApplicationManager.idl file.

Detailed Proposal

3.1.3.3.1.1 ApplicationManager – Update Section

3.1.3.3.1.1.5.4 disconnectPorts 3.1.3.3.1.1.5.4.1 Brief Rationale

The *disconnectPorts* operation is used to disconnect the application external ports as defined in the associated SAD (ApplicationManagerComponent's profile ~~attribute~~).

Detailed Proposal

3.1.3.3.1.3 ApplicationFactory

3.1.3.3.1.3.2 UML

Update Figure 3-22: ApplicationFactory Interface UML

Remove 3.1.3.3.1.3.4.2 softwareProfile

3.1.3.3.1.3.5.1.3 Behavior

The create operation locates candidate DomainManagerComponents, ApplicationFactoryComponents or DeviceComponents capable of deploying application software modules based upon information in its associated SAD (~~softwareProfile attribute~~ ApplicationFactoryComponent's profile).

Update Appendix C and CFApplicationFactory.idl file.

Detailed Proposal

ApplicationManager Interface

3.1.3.3.1.1.5.1 releaseObject

3.1.3.3.1.1.5.1.3 Behavior

The ApplicationManager::releaseObject operation shall send a ~~DomainManagementObjectRemoved~~ComponentChangedEventType event to the Outgoing Domain Management event channel upon successful release of an application. For this event,

1. The producerId is the identifier attribute of the releasing ApplicationManagerComponent.
2. The componentChange is REMOVED
3. The *domainComponent is the released ApplicationManagerComponent 's ComponentType.*

Detailed Proposal

ApplicationFactory Interface

3.1.3.3.1.3.5.1.2 Synopsis

Create operation Return type change to ComponentType

ComponentType create (in string name, in Properties initConfiguration, in DeviceAssignmentSequence deviceAssignments, in Properties deploymentDependencies) raises (CreateApplicationError, CreateApplicationRequestError, InvalidInitConfiguration);

3.1.3.3.1.3.5.1.4 Returns

The *create* operation shall return the created ApplicationManagerComponent's ComponentType for the created application when the application is successfully created

For the returned ComponentType, its profile, componentObject, and type fields are as specified in 3.1.3.1.3.17 ComponentType.

For the returned ComponentType's identifier field is the input name parameter.

The returned ComponentType's providesPorts field is the ApplicationManagerComponent's external provides ports.

Update Appendix C and CFDomainInstallation IDL file

Detailed Proposal

3.1.3.3.1.3.5.1.5 Exceptions

Add Exception requirement

SCAXXX The create operation shall raise the CreateApplicationError exception when an ApplicationManagerComponent already exist in the system for which the ComponentType's identifier attribute value is the same as the input name parameter.

Detailed Proposal

Domain Manager Interface

- 3.1.3.3.1.4.3 Types
  - Remove ApplicationSeq and ApplicationFactorySeq.
- 3.1.3.3.1.4.4 Attributes
  - Applications
    - readonly attribute Components applications;
  - ApplicationFactories
    - readonly attribute Components applicationFactories;

Update DomainManager Interface Figure

Update Appendix C and CFDomainManager IDL file

Detailed Proposal

Domain Installation Interface

- Update Figure
- Change InstallApplication
- 3.1.3.3.1.5.5.1.2 Synopsis
  - ComponentType installApplication (in string profileFileName) raises (InvalidProfile, InvalidFileName, ApplicationInstallationError, ApplicationAlreadyInstalled);
- **3.1.3.3.1.5.5.1.3 Behavior**
  - The installApplication operation shall send a ~~DomainManagementObjectAdded~~ComponentChangedEventType event to the Outgoing Domain Management event channel, upon successful installation of an application. For this event,
    1. The producerId is the identifier attribute of the DomainManagerComponent.
    2. The componentChange is ADDED
    3. The domainComponent is the returned ComponentType.

Driving the future of radio communications and systems worldwide

Detailed Proposal

Domain Installation Interface, cont'd

3.1.3.3.1.5.5.1.4 Returns

This operation shall return~~s~~ the installed ApplicationFactoryComponent's ComponentType ~~which includes the name that is required for uninstallApplication (this is the softwareassembly element name attribute of the ApplicationFactory's SAD file)~~. For the returned ComponentType, its profile, componentObject, and type fields are as specified in 3.1.3.1.3.17 ComponentType. The returned ComponentType's identifier field is the input profileFileName's element name attribute.

Update Appendix C and CFDomainInstallation IDL file

Driving the future of radio communications and systems worldwide

Detailed Proposal

Domain Installation Interface, cont'd

Change uninstallApplication operation

3.1.3.3.1.5.5.2.3 Behavior

The uninstallApplication operation shall send a ~~DomainManagementObjectRemoved~~ComponentChangeEventType event to the Outgoing Domain Management event channel, upon the successful uninstallation of an application. For this event,

1. The producerId is the identifier attribute of the DomainManagerComponent.
2. The componentChange is REMOVED.
3. The domainComponent is the uninstalled ApplicationFactoryComponent's ComponentType.

Detailed Proposal

3.1.3.3.1.10 ReleasableManager

3.1.3.3.1.10.5.1.3 Behavior

SCA152 The *shutdown* operation shall perform a *releaseObject* on all of the manager's registered components that were created as specified in the profile ~~attribute~~ that supports the *LifeCycle* interface.

SCA153 The *shutdown* operation shall terminate the execution of each component that was created as specified in the profile ~~attribute~~ after they have unregistered with the manager.

Detailed Proposal

3.1.3.3.2.1.3 ApplicationManagerComponent Semantics

SCA163 The ApplicationManagerComponent shall raise *configure* operation InvalidConfiguration exception when the input configProperties parameter contains unknown properties. A property is considered unknown if it can't be recognized by a component designated by the ApplicationManagerComponent profile~~ attribute~~'s *assemblycontroller* element.

Detailed Proposal

3.1.3.3.2.2.3 Application Factory Component Semantics

The create operation shall send a ~~DomainManagementObjectAdded~~ComponentChangeEventType event to the Outgoing Domain Management event channel upon successful creation of an application.

For this event:

1. The producerId is the identifier attribute of the ApplicationFactoryComponent.
2. The componentChange is ADDED.
3. The domainComponent is returned ApplicationManagerComponent's ComponentType.

Detailed Proposal

3.1.3.3.2.3.3 Domain Manager Component Semantics

Registering Event

The registerComponent operation shall send a ~~DomainManagementObjectAdded~~ComponentChangeEventType event to the Outgoing Domain Management event channel, upon successful registration of a component. For this event,

1. The producerId is the identifier attribute of the DomainManagerComponent.
2. The componentChange is ADDED.
3. The domainComponent is the input registering component's ComponentType.

Detailed Proposal

3.1.3.3.2.3.3 Domain Manager Component Semantics

Unregistering Event

The unregisterComponent operation shall send a ~~DomainManagementObjectRemoved~~ComponentChangeEventType event to the Outgoing Domain Management event channel, upon successful unregistration of a component. For this event,

1. The producerId is the identifier attribute of the DomainManagerComponent.
2. The componentChange is REMOVED.
3. The domainComponent is the unregistered component's ComponentType.

Detailed Proposal

3.1.3.4.2.1.3 DeviceComponent Semantics

SCA458 A child DeviceComponent shall add itself to a parent device using the executable Composite Device IOR and DEVICE_ID parameters per 3.1.3.3.2.4.3. SCA299 The values associated with the parameters (~~PROFILE_NAME, ~~COMPOSITE_DEVICE_IOR, and DEVICE_ID) as described in 3.1.3.3.2.4.3 shall be used to set the DeviceComponent's ~~softwareProfile, ~~compositeDevice~~,~~ and identifier attributes, respectively.

3.1.3.4.2.2.3 LoadableDeviceComponent Semantics

SCA306 The *load* operation shall support the load types as stated in the LoadableDeviceComponent's ~~software ~~profile LoadType allocation properties. SCA307 When a LoadType is not defined for the LoadableDeviceComponent, the *load* operation shall support all code types. The type field of the registering component is a LOADABLE_DEVICE_COMPONENT.

3.1.3.5.2.5.3 PlatformComponentFactoryComponent Semantics

A ~~software ~~DCD profile specifies which PlatformComponentFactoryComponents are to be used by the DeviceManagerComponent

Driving the future of radio communications and systems worldwide

3.1.3.4.1.3 DeviceAttributes – Update Section

3.1.3.4.1.3.1 Description

The *DeviceAttributes* interface defines attributes for any logical device in the domain. A logical device may provide the following attributes:

~~1. Software Profile Attribute - Either the filename of the SPD or the raw SPD that defines the logical device capabilities (data/command uses and provides ports, configure and query properties, capacity properties, status properties, etc.), which could be a subset of the hardware device's capabilities;~~

~~2~~1. Operational State Management - This information describes the operational states of the device.

3.1.3.4.1.3.2 UML

Update Figure 3-39

Remove Section 3.1.3.4.1.3.4.2 softwareProfile

Update Appendix C and CFDeviceAttributes.idl file.

Driving the future of radio communications and systems worldwide

Detailed Proposal

3.1.3.4.1.2 CapacityManagement

3.1.3.4.1.2.1 Description

2. Capacity Operations - In order to use a device, certain capacities (e.g., memory, performance, etc.) are obtained from the device. A device may have multiple capacities which need to be allocated, since each device has its own unique capacity model which is described in the associated ~~software~~ SPD profile.