Proposal for SCA v4.1 Push Registration - Allocation Properties

Specification Changes

Document WINNF-14-R-0012

Version V1.0.0

16 September 2014

Contents

Main Specifications Changes

2.2.4 Core Framework Control Architecture

remove the ManagerRegistry and FullManagerRegistry boxes from Framework Control Interfaces

Figure 3-2: Core Framework IDL Relationships

remove the ManagerRegistry and FullManagerRegistry boxes from diagram

Section 3.1.3.1.2.1.4 Constraints

Add

A ComponentBase registering to a ComponentRegistry or created by a ComponentFactory shall set the following ComponentType fields: identifier, type, componentObject and providePorts.

3.1.3.1.3.16 ComponentEnumType

Change From

The ComponentEnumType enumeration defines the basic type of a component. The APPLICATION_COMPONENT field is a component which is launched as part of a Software Assembly. The DEVICE_COMPONENT field is a ComponentBaseDevice launched by a DeviceManagerComponent. The CF_SERVICE_COMPONENT field is a CF_ServiceComponent launched by a DeviceManagerComponent that the framework can manage through the CF based interfaces. The NON_CF_SERVICE_COMPONENT is a ServiceComponent launched by a DeviceManagerComponent that could implement possibly any interface (e.g. Log, FileSystem, etc.). The FRAMEWORK_COMPONENT is a DeviceManagerComponent, DomainManagerComponent, ApplicationManagerComponent, or ApplicationFactoryComponent.

To

The ComponentEnumType enumeration defines the basic type of a component. The MANAGEABLE_APPLICATION_COMPONENT value is used to identify a ManageableApplicationComponent which is launched by an ApplicationFactoryComponent. The COMPONENT_FACTORY value is used to identify a ComponentFactoryComponent which is launched by an ApplicationFactoryComponent or launched by a DeviceManagerComponent..

The RESOURCE and the RESOURCE_FACTORY values are used to identify a version 2.2.2 component which is launched by an ApplicationFactoryComponent. The APPLICATION_FACTORY value is used to identify an ApplicationFactoryComponent which is created by a DomainManagerComponent during application installation. The APPLICATION_MANAGER value is used to identify an ApplicationManagerComponent created by an ApplicationFactoryComponent. The DEVICE value is used to identify a DeviceComponent launched by a DeviceManagerComponent. The LOADABLE_DEVICE value is used to identify a LoadableDeviceComponent launched by a DeviceManagerComponent. The EXECUTABLE_DEVICE value is used to identify an ExecutableDeviceComponent launched by a DeviceManagerComponent. The MANAGEABLE_SERVICE is used to identify a ManageableServiceComponent launched by a DeviceManagerComponent that the framework can manage through the CF based interfaces. The LOG, FILESYSTEM, EVENT_SERVICE are values used to identify a ServiceComponent launched by a DeviceManagerComponent that the framework can use through known CF interfaces. The SERVICE is used to identify a ServiceComponent launched by a DeviceManagerComponent that is not managed by CF. The DEVICE_MANAGER is used to identify a DeviceManagerComponent registered with the DomainManagerComponent. The DOMAIN_MANAGER is used to identify a DomainManagerComponent.

Change From

enum ComponentEnumType

{

APPLICATION_COMPONENT,

DEVICE_COMPONENT,

CF_SERVICE_COMPONENT,

NON_CF_SERVICE_COMPONENT,

FRAMEWORK_COMPONENT

};

To

enum ComponentEnumType

{

MANAGEABLE_APPLICATION_COMPONENT, COMPONENT_FACTORY, RESOURCE, RESOURCE_FACTORY, APPLICATION_FACTORY, APPLICATION_MANAGER, DEVICE, LOADABLE_DEVICE, EXECUTABLE_DEVICE, LOG, FILESYSTEM, EVENT_SERVICE, SERVICE, MANAGEABLE_SERVICE, DEVICE_MANAGER,DOMAIN_MANAGER

};

3.1.3.1.3.17 ComponentType

Change From

The ComponentType structure defines the basic elements of a component. The identifier field is the id of the component as specified through execparams. The softwareProfile field is either the component's SPD filename or the SPD itself. The type field is the type of component. The componentObject field is the object reference of the component. The providesPorts field is a sequence of static ports provided by the component.

To

The ComponentType structure defines the basic elements of a component. The identifier field is the id of the component as specified through creation parameters. The managerIdentifier field is the identifier of the component's manager. The profile field is either the component's profile filename or the profile itself, the DomainManagerComponent uses a DMD profile, the ApplicationFactoryComponent and the ApplicationManagerComponent uses a SAD profile, the DeviceManagerComponent uses a DCD profile and all other component uses SPD profile. The type field is the equivalent value of the SCD componenttype element described in Appendix D paragraph D-1.9.1.2. The componentObject field is the object reference of the component. The supportedInterface field is a sequence of interface identifiers that are found in the SCD componentrepid element and the repid attribute of all the supportedinteraces elements described in Appendix D paragraph D-1.9.1.1 and D-1.9.1.3.1 repectively. The providesPorts field is a sequence of static ports provided by the component. The specializeInfo field is a variant used only for some components to store a component specialized information that is specific to component type.

Change From

struct ComponentType

{

string identifier;

string softwareProfile;

ComponentEnumType type;

Object componentObject;

Ports providesPorts;

};

to

```
struct ComponentType
{
string  identifier;
string    managerIdentifier;
string  profile;
ComponentEnumType type;
Object componentObject;
CF:StringSequence supportedInterfaces;
CF::Ports providesPorts;
Any    specializedInfo; // component specific
};
```

3.1.3.1.3.19 ManagerType

remove section

3.1.3.1.3.20 RegisterError

Change From

The RegisterError exception indicates that an internal error has occurred to prevent the ComponentRegistry **or ManagerRegistry** interface registration operations from successful completion.

To

The RegisterError exception indicates that an internal error has occurred to prevent the ComponentRegistry interface registration operations from successful completion.

3.1.3.1.3.21 UnregisterError

Change From

The UnregisterError exception indicates that an internal error has occurred to prevent the

FullComponentRegistry **or FullManagerRegistry** interface unregister operations from successful

completion.

To

The UnregisterError exception indicates that an internal error has occurred to prevent the

FullComponentRegistry interface unregister operations from successful

completion.

3.1.3.1.3.25 PropertyActionType

Add new enum PropertyActionType

This enum is used to represent an allocation property type value that is equivalent to the action value of the action element described in Appendix D paragraph D.4.1.1.7

enum PropertyActionType

{

CF_EQ, CF_NE, CF_GT, CF_GE,CF_LT, CF_LE, CF_EXTERNAL

};

3.1.3.1.3.26 PropertyType

Add new enum PropertyType

This enum is used to represent an allocation property action value that is equivalent to the attribute type value of the simple element or simplesequence element described in Appendix D paragraph D.4.1.1 and paragraph D 4.1.2 respectively.

enum PropertyType

{

CF_BOOLEAN, CF_CHAR, CF_DOUBLE, CF_FLOAT, CF_SHORT, CF_LONG,

CF_OBJREF,CF_OCTET,CF_STRING,CF_USHORT,CF_ULONG

};

3.1.3.1.3.27 AllocationPropertyType

Add new struct AllocationPropertyType

The AllocationPropertyType structure defines the basic information to store a simple allocation property or a simplesequence allocation property. The field matches the definitions found in Appendix D paragraph D.4.1.1 and paragraph D 4.1.2 respectively. The id field is the id attribute of a property. The values field is a sequence of string values that can store a simple property value or a simplesequence property values. The action field is enumeration to store the action value that is equivalent to the action element value. The type field is an enumeration to store the type value that is equivalent to the attribute type value of the simple element or simplesequence element

```
struct AllocationPropertyType
{
  string id;
  CF::StringSequence values;
  CF::PropertyActionType action;
  CF::PropertyType type;
}
```

3.1.3.1.3.28 AllocationProperties

Add new sequence AllocationProperties

The AllocationProperties type defines a sequence of AllocationPropertyType structures.

```
typedef sequence< AllocationPropertyType>  AllocationProperties;
```

3.1.3.1.3.29 PlatformComponentInfo

Add new struct PlatformComponentInfo

The PlatformComponentInfo structure defines a type for information needed to register a PlatformComponent. The allocationProperties field is the sequence of allocation properties of the PlatformComponent.

```
struct PlatformComponentInfo
{
  CF::AllocationProperties allocationProperties;
 }
```

3.1.3.1.3.30 DeviceManagerInfo

Add new struct DeviceManagerInfo

The DeviceManagerInfo structure defines the specific information of a device manager component. The fileSys field is the file system used by this manager component.

The registeredComponents field is a sequence of components that have registered with this manager component.

```
struct DeviceManagerInfo
{
  CF::FileSystem fileSys;
  CF::Components registeredComponents;
}
```

3.1.3.3 Framework Control

3.1.3.3.1 Interfaces

remove “ManagerRegistry” and “FullManagerRegistry”

Figure 3-24: DomainManager Interface UML

Change managers : ManagerSeq to manager: Components

3.1.3.3.1.4.3.1 ManagerSeq

delete section

3.1.3.3.1.4.4.1 managers

Change From

readonly attribute ManagerSeq managers;

to

readonly attribute Components managers;

3.1.3.3.1.11 ManagerRegistry

delete section

3.1.3.3.1.12 FullManagerRegistry

delete section

Figure 3-38: DomainManagerComponent UML

remove the ManagerRegistery box from diagram

Change managers : ManagerSeq to manager: Components

3.1.3.3.2.3 ApplicationFactoryComponent

3.1.3.3.2.3.3 Semantics

SCAXXX The ApplicationFactoryComponent's identifier shall be assigned to the ComponentType managerIdentifier field for any ApplicationComponent.

- Should be mapped to Interrogable UOF in Appendix F attachment 1

SCAXXXThe registerComponent operation shall set the following ComponentType fields :profile and supportedInterface for any ApplicationComponent instantiated by the ApplicationFactoryComponent.

- Should be mapped to Interrogable UOF in Appendix F attachment 1

SCAXXX The registerComponent operation shall add the registeringComponent to the

- Should be mapped to Interrogable UOF in Appendix F attachment 1

ApplicationDeploymentData's registeredComponents attribute.

Note: All requirements regarding the information in the ApplicationDeploymentData is missing from the spec

3.1.3.3.2.4 DomainManagerComponent

3.1.3.3.2.4.2 Associations

remove the managerRegistry description

remove the word "platform" from the componentRegistry description

Modify and move SCA144 to 3.1.3.3.2.4.3 :

3.1.3.3.2.4.3 Semantics

SCA144 The registerComponent operation from a DeviceManagerComponent shall register the registeredComponents provided in the DeviceManagerInfo struct assigned to the ComponentType specializedInfo attribute.

Change and move SCA149 to 3.1.3.3.2.4.3

SCA149 The unregisterComponent operation from a DeviceManagerComponent shall unregister all components associated with the manager that is being unregistered.

Change

SCA201 The **registerManager** operation shall establish any connections **for the DeviceManagerComponent indicated by the input registeringManager parameter,** …
to

SCA201 The **registerComponent** operation **for a DeviceManagerComponent** shall establish any connections which are specified in the connections element of…

Main Specifications Changes

Replace word “registerManager” by “registerComponent”

SCA202 For connections established for an Event Service's event channel, the **registerComponent** operation shall connect a CosEventComm::PushConsumer or CosEventComm::PushSupplier object to the event channel as specified in the DCD's domainfinder element.

SCA203 If the event channel does not exist, the **registerComponent** operation shall create the event channel.

SCA204 The **registerComponent** operation from a DeviceManagerComponent shall mount the DeviceManagerComponent's FileSystemComponent to the DomainManagerComponent's FileManagerComponent.

Remove

SCA206 since it is a duplicate of SCA191

SCA207 since it is a duplicate of SCA193

Note that the successful registerDeviceManager event requirement was missing but it is covered by SCA 189

Change

The DomainManagerComponent associates the input DeviceManagerComponent's registered components with the DeviceManagerComponent in order to support the **unregisterManager** operation.

to

The DomainManagerComponent associates the input DeviceManagerComponent's registered components with the DeviceManagerComponent in order to support the **unregisterComponent** operation of a DeviceManagerComponent.

Change

SCA208 The *unregisterManager* operation shall …

to

SCA208 The ***unregisterComponent*** operation for DeviceManagerComponent shall …

Remove

SCA209 since it is a duplicate of SCA199

Change

The unregisterManager operation may destroy the Event Service channel when no more consumers and producers are connected to it.

To

The **unregisterComponent** operation may destroy the Event Service channel when no more consumers and producers are connected to it.

Change

SCA210 The unregisterManager operation shall unmount all DeviceManagerComponent's file systems from its FileManagerComponent.

to

SCA210 The unregisterComponent operation for a DeviceManagerComponent shall unmount all DeviceManagerComponent's file systems from its FileManagerComponent.

Remove

SCA211 since it is a duplicate of SCA195

SCA212 since it is a duplicate of SCA197

SCA213 since it is a duplicate of SCA196

Figure 3-39: DeviceManagerComponent UML

remove ManagerRegistery box from diagram

3.1.3.3.2.5 DeviceManagerComponent

3.1.3.3.2.5.2 Associations

Remove

the registrar: description

Replace word “ManagerRegistry” by “ComponentRegistry”

SCA216 A DeviceManagerComponent upon start up shall register with a DomainManagerComponent via the **ComponentRegistry** interface.

Add after sentence SCA210

SCAXXX The DeviceManagerComponent shall set it’s own ComponentType fields :profile and supportedInterface before registering with the DomainManagerComponent.

SCAXX The DeviceManagerComponent shall insert a DeviceManagerInfo structure into it’s ComponentType specializedInfo field. The fileSys field is the DeviceManager’s fileSystem. The registeredComponents field is set to the same value as the DeviceManagerComponent’s registeredComponents attribute. For every platform component that has allocation properties in their profile, the allocation property information is added to PlatformComponentInfo allocationProps sequence, then the PlatformComponentInfo is inserted in the ComponentType specializedInfo field.

SCAXXXThe DeviceManagerComponent shall set the following ComponentType fields :profile and supportedInterface for any PlatformComponent instantiated by the DeviceManagerComponent .

Add to paragraph SCA230

SCAXXX The DeviceManagerComponent identifier shall be assigned to the ComponentType managerIdentifier field for platform component registration with the DomainManager.

3.1.3.4.2.1 ComponentBaseDevice

3.1.3.4.2.1.3 Semantics

SCA298 A ComponentBaseDevice shall register with its **launching** DeviceManagerComponent ...

to

SCA298 A ComponentBaseDevice shall register with its **associated** DeviceManagerComponent ...

Appendix C Specification Changes (IDL)

C.7.1.2 CFCommonTypes IDL

Change From

```
struct ComponentType
{
string identifier;
string softwareProfile;
ComponentEnumType type;
Object componentObject;
Ports providesPorts;
};
```

to

```
struct ComponentType
{
string  identifier;
string    managerIdentifier;
string  profile;
ComponentEnumType type;
Object componentObject;
CF:StringSequence supportedInterfaces;
```

CF::Ports providesPorts;

Any specializedInfo; // component specific

};

Change from

Struct ComponentEnumType

{

APPLICATION_COMPONENT, DEVICE_COMPONENT, CF_SERVICE_COMPONENT, NON_CF_SERVICE_COMPONENT, and FRAMEWORK_COMPONENT
};

to

Struct ComponentEnumType

{

MANAGEABLE_APPLICATION_COMPONENT, APPLICATION_COMPONENT, COMPONENT_FACTORY, RESOURCE, RESOURCE_FACTORY, APPLICATION_FACTORY, APPLICATION_MANAGER, DEVICE, LOADABLE_DEVICE, EXECUTABLE_DEVICE, LOG, FILESYSTEM, EVENT_SERVICE, SERVICE, MANAGEABLE_SERVICE, DEVICE_MANAGER,DOMAIN_MANAGER

};

C.7.1.5 CFPlatformComponentInfo IDL

Add new Section

Add new enum PropertyActionType

enum PropertyActionType

{

CF_EQ, CF_NE, CF_GT, CF_GE,CF_LT, CF_LE, CF_EXTERNAL

};

Add new enum PropertyType

enum PropertyType

```
{
CF_BOOLEAN, CF_CHAR, CF_DOUBLE, CF_FLOAT, CF_SHORT, CF_LONG,
CF_OBJREF,CF_OCTET,CF_STRING,CF_USHORT,CF_ULONG
};
```

Add new struct AllocationPropertyType

```
struct AllocationPropertyType
{
   string id;
  CF::StringSequence values;
   CF::PropertyActionType action;
   CF::PropertyType type;
}
```

Add new Sequence

```
typedef sequence< AllocationPropertyType>  AllocationProperties;


struct PlatformComponentInfo
{
  CF::AllocationProperties allocationProperties;
 };
```

C.7.1.6 CFDeviceManagerInfo IDL

Add new Section

```
struct DeviceManagerInfo
{
 CF::FileSystem fileSys;
```

CF::Components registeredComponents;

}

C.7.4.7 CFDomainManager IDL

Remove

/* This type defines an unbounded sequence of DeviceManagers. */

typedef sequence <ManagerType> ManagerSeq;

Change From

readonly attribute CF::DomainManager::ManagerSeq managers;

To

readonly attribute CF::Components managers;

C.7.4.10 CFFullManagerRegistry IDL

Delete section

C.7.4.11 CFManagerRegistry IDL

Delete section

Appendix D Specification Changes (DTDs)

D-1.9.1.1 componentrepid

Change From:

The componentrepid uniquely identifies the interface that the component is implementing. The componentrepid may be referred to by the componentfeatures element. **The componentrepid is derived from the Resource, Device, LoadableDevice, ExecutableDevice, ComponentFactory interface or represents a ServiceComponent.**

.

Change To:

The componentrepid uniquely identifies the **principal** interface that the component is implementing. The componentrepid may be referred to by the componentfeatures element. The componentrepid is **the IDL repository Id of the component's principal interface. (see CORBA Interfaces, Version 3.2 [1] section 14.7.1 IDL Format)**

Note: principal interface is the terminology used in CORBA Interfaces, Version 3.2 [1]

D-1.9.1.2 componenttype

change the types from

APPLICATION_COMPONENT, DEVICE_COMPONENT, CF_SERVICE_COMPONENT, NON_CF_SERVICE_COMPONENT, and FRAMEWORK_COMPONENT
to

applicationcomponent, manageableapplicationcomponent, componentfactory , applicationfactory, applicationmanager, device, loadabledevice, executabledevice log, filesystem, eventservice, service, manageableservice, devicemanager, domainmanager

D-1.9.1.3 componentfeatures

Change From :

The componentfeatures element (see Figure 17) is used to describe a component with respect to the components that i**t inherits from, the interfaces the component supports, its provides and uses ports**. **If a component extends any of the following interfaces, Resource, ComponentFactory, or Device,**

LoadableDevice, ExecutableDevice ,then all the inherited interfaces (e.g., Resource) are depicted as supportsinterface elements.

Change To :

The componentfeatures element (see Figure 17) is used to describe a component with respect to **the interfaces that are inherited directly or indirectly by the component's principal interface,** its provides and uses ports

D-1.9.1.3.1 supportsinterface

Change From:

The supportsinterface element is used to identify an interface definition that the component supports. **These interfaces are distinct interfaces that were inherited by the component's specific interface.** One can widen the component's interface to be a supportsinterface. The repid is used to refer to the interface element (see interfaces section D-1.9.1.4)

Change To:

The supportsinterface element is used to identify an interface definition that the component supports. These interfaces are **all the interfaces that were inherited directly or indirectly by the component's principal interface described in the componentrepid element**.One can widen the component's interface to be a supportsinterface. **The repid is the IDL repository Id as defined in CORBA Interfaces, Version 3.2 [1] section 14.7.1 IDL Format.** The repid is used to refer to the interface element (see interfaces section D-1.9.1.4)

D-1.9.1.4 interfaces

Change From :

The interface element describes an interface that the component, either **directly or through inheritance**, provides, uses**, or supports**. The name attribute is the character-based non-qualified name of the interface. The repid attribute is the unique repository id of the interface. The repid is also used to reference an interface element elsewhere in the SCD, for example from the inheritsinterface element

Change To:

The interface element describes an interface that the component, either **inherits directly or indirectly**, provides,or uses. The name attribute is the character-based non-qualified name of the interface. The repid attribute is the unique **IDL** repository id of the interface **as defined in CORBA Interfaces, Version 3.2 [1] section 14.7.1 IDL Format.**. The repid is also used to reference an interface element elsewhere in the SCD, for example from the inheritsinterface element

Remove

For ServiceComponents the inheritsinterface element is not expected to contain a value.

D-1.8.1 properties

D-1.8.1.1.6 kind

Change From

3. allocation, which is used in the allocateCapacity and deallocateCapacity operations of the Device interface. The ApplicationFactoryComponent and DeviceManagerComponent will use the simple properties of kindtype allocation to build the input capacities parameter to the allocateCapacity operation that is invoked on device components during application creation, when the action element of those properties is external. The application factory and device manager manage simple properties of kindtype allocation when the action is not external.

Change To

3. allocation, … The ApplicationFactoryComponent uses a component's dependency simple properties of kindtype allocation and action external to build the input capacities parameter to the allocateCapacity operation that is invoked on device components during application creation. The ApplicationFactoryComponent uses a component's dependency simple properties of kindtype allocation with action not external to perform the dependency check with a device component during application creation. A DeviceManagerComponent handles the allocation properties the same way as the ApplicationFactoryManager when deploying a platform component onto a device component. A device component can only use a simple property for an external allocation property but may use a simple property or a simple sequence property for an allocation property that is not external. When a device uses an allocation simple sequence, the dependency check is successful when one of the sequence value pass the check.