Joint Tactical Networking Center Test and Evaluation Laboratory

Software Communications Architecture 2.2.2
Manual Operating Environment Software Test Description v3.3A

Appendix D
SCA Extensions Requirements Test Procedures

13 April 2022

Prepared for:
Joint Tactical Networking Center
33000 Nixie Way, Bldg 50, Suite 339
San Diego, CA 92147-5110

Prepared by:
Joint Tactical Networking Center Test and Evaluation Laboratory

TABLE OF CONTENTS

APPENDIX D SCA EXTENSIONS REQUIREMENTS TEST PROCEDURES

This Appendix contains the test case procedures for the SCA Extensions requirements. The requirements tested here come from the SCA Extensions document dated 22 December 2006. The test case procedures include the test case name, test objective, preconditions, parameters, test description, the name of the related automated test name (if relevant), and the test requirement(s) being tested.

The procedures, placed in a table, will use the test description as the 'headers' followed by the detailed manual test steps. The test description provides 'what' is being tested – not the 'how'. Each row contains the step(s) of what the tester needs to do (such as examine, verify, locate, etc.) to complete the test description. The five columns of the table seen in the manual test steps are Steps, Expected Results, Actual Results, Comments and Test Results:

1. **Column 1 (Steps) -** Contains what the tester is to do to verify the requirement(s).
2. **Column 2 (Expected Results) -** Contains the expected consequence that results from the action of the first column.
3. **Column 3 (Actual Results) -** Provides the location where the tester can record the results when Column 1 is performed. Additional space is provided in the Test Recording Log.
4. **Column 4 (Comments) -** Contains any additional information to aid the tester in the running of the test.
5. **Column 5 (Test Result) -** Provides the location where the tester records the results of the manual steps. The results would be Pass, Fail, Untested or N/A.

The definitions for the various test results are as follows:

- **N/A**: The requirement is not applicable to this component.
- **Untested**: The test could not be completed because required information is missing. If the test is a final test and the developer cannot provide the missing information, this becomes a failure.
- **Fail**: The expected results of a step were not met.
- **Pass**: The expected results were met. This step passes. If it is the last step for a requirement and all the steps for that requirement passed then the requirement passes.

To aid in the execution of the test and provide test artifacts, each test case has a Test Recording Log. The Test Recording Log is used to record test information and is very helpful when there are many items that must be verified by the tester. For example, if a manual step requires the tester to locate several properties files (PRF), and then the tester can write down the names of all the files used in the test.

At the end of each test case, there is a summary page listing each requirement that was verified through the course of the test case. The tester is encouraged to provide the results of each requirement, which will serve as a quick summary of the status for each requirement. This is also where the participants of the test will sign their names and indicate the date that the test was run.

To perform a test of an application, record the names of the operating system (OS), the operating environment (OE), the CORBA Object Request Broker (ORB), and the Core Framework (CF).

- OS _____
- OE _____
- ORB _____
- CF _____

D.1 OE_TC_176 - DomainManager :: registerService registers non-SCA services

Test Case Number: OE_TC_176

DomainManager::registerService

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0771	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.
OE0772	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-46 to 3-47, Section 3.1.3.2.3.6.7.3
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 1, Section 2.1 Page 7, Section 6.10
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-20

Test Objective

This test case verifies OE0771 and OE0772. The objective of this test is to verify that upon successful service registration of a non-SCA service with an identifier and type in the 'identifier\type" format, the *registerService* operation makes the value provided in the "identifier" portion and the value provided in the "type" portion, accessible through the *domainfinder servicename* and *servicetype* mechanisms, respectively.

The SCA Extensions add two types to the attribute definition of the *domainfinder* element. Using either of these two types automatically indicates that the service is non-SCA. The 'identifier\type" format is exclusively used within the SCA Extensions

document and is specifically used for registering non-SCA services. These requirements are saying that the registration process stores the type and identifier within the domainfinder mechanism and implied in the requirement is that the identifier or type may be used to retrieve the registered service once it has been registered.

Note: The requirements and this test case use the term **mechanism** to mean a collection of software in support of the named functionality. Being more specific may lead to a programming language dependent discussion.

Definitions:

"identifier" as used above is that portion of the input name that exists before the first back slash character in the "identifier\type" format. The identifier corresponds to the DCD *componentinstantiation* usagename element.

"type" as used above is that portion of the input name that exists after the first back slash character in the "identifier\type" format. It must correspond to the *componentrepid* found in the SCD file.

Places to Verify

DomainManager and the support software for the *domainfinder*.

IDL References

Operations

void registerService (in Object registeringService, in DeviceManager registeredDeviceMgr, in string name)

raises (InvalidObjectReference, DeviceManagerNotRegistered, RegisterError);

Preconditions

- The DomainManager source code files are available.
- The source code files, which support the domainfinder mechanism are available.
- The Domain Profile files are available.
- The SCA Extensions are implemented.

Test Description

For each domainManager within the OE under test, perform the following steps:

A. Verify that there are "servicename" or "servicetype" attributes in the *domainfinder* element of the DCD file. (OE0771, OE0772)
   1. **N/A**: No servicename or servicetype attributes can be found in the DCD file's domainfinder element.

B. Locate the source code for the DomainManager *registerService* operation. (OE0771, OE0772)
   1. **Pass**: The source code for the DomainManager *registerService* operation is found.
   2. **Untested**: The source code for the DomainManager *registerService* operation is not found.

For each *registerService* operation found

C. Verify that the *registerService* operation determines when the "identifier\type" format is being used. (OE0771, OE0772)
   1. **Pass:** The *registerService* operation determines that the "identifier\type" format is being used.
   2. **Fail:** The *registerService* operation does not determine that the "identifier\type" format is being used.

The code supporting the next two steps must only be executed upon successful registration of the service with the "identifier\type" format as the input name.

D. Verify that the *registerService* operation, associates the identifier portion of the input name parameter (in the "identifier/type" format) to the reference to the service in the DomainFinder. (OE0771)
   1. **Pass:** The *registerService* operation associates the identifier portion of the input name parameter to the reference to the service in the DomainFinder.
   2. **Fail:** The *registerService* operation does not associate the identifier portion of the input name parameter to the reference to the service in the DomainFinder.

E. Verify that the *registerService* operation associates the type portion of the input name parameter (in the "identifier/type" format) to the reference to the service in the DomainFinder. (OE0772)
   1. **Pass:** The *registerService* operation associates the type portion of the input name parameter to the reference to the service in the DomainFinder.
   2. **Fail:** The *registerService* operation does not associate the type portion of the input name parameter to the reference to the service in the DomainFinder.

F. Locate the source code that supports the *domainfinder* mechanism. (OE0771, OE0772)
   1. **Pass**: The source code that supports the *domainfinder* mechanism is found.
   2. **Untested**: The source code that supports the *domainfinder* mechanism is not found.

G. Verify that a get object reference operation within the *domainfinder* mechanism uses "servicename" as one of its input object types and an input object name in searching for an object. (OE0771)

1. **Pass:** The get object reference operation within the *domainfinder* mechanism uses "servicename" as one of its input object types and an input object name in searching for an object.
2. **Fail:** The get object reference operation within the *domainfinder* mechanism does not use "servicename" as one of its input object types and/or an input object name in searching for an object.

H. Verify that a get object reference operation within the *domainfinder* mechanism uses "servicetype" as one of its input object types and an input object name in searching for an object. (OE0772)
   1. **Pass:** The get object reference operation within the *domainfinder* mechanism uses "servicetype" as one of its input object types and an input object name in searching for an object.
   2. **Fail:** The get object reference operation within the *domainfinder* mechanism does not use "servicetype" as one of its input object types and/or an input object name in searching for an object.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Verify that there are "servicename" or "servicetype" attributes in the *domainfinder* element of the DCD file. (OE0771, OE0772)				
1. Perform a search on all DCD files provided by the developer for the OE.	Find the DCD files of the OE.			
2. Verify that there are "servicename" or "servicetype" attributes in the *domainfinder* element of the DCD file.	**N/A**: No "servicename" or "servicetype" attributes in the *domainfinder* element of the DCD file can be found. (OE0771, OE0772)			
For each DomainManager within the OE under test				
B. Locate the source code for the DomainManager *registerService* operation. (OE0771, OE0772)				

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Perform a search on all source code provided by the developer for the implementation of the *DomainManager registerService* operation and record the source code file name.	**Pass:** The source code for the *DomainManager registerService* operation is found. (OE0771, OE0772) **Untested**: The source code for the *DomainManager registerService* operation is not found. (OE0771, OE0772)			
For each *registerService* operation found				
C. Verify that the *registerService* operation determines when the "identifier\type" format is being used. (OE0771, OE0772)				
4. Verify that the *registerService* operation processes the "identifier\type" format that is being used in the input name parameter.	**Pass:** The *registerService* operation processes the "identifier\type" format that is being used. (OE0771, OE0772) **Fail:** The *registerService* operation does not process the "identifier\type" format that is being used. (OE0771, OE0772)		The code should attempt to parse the input name parameter, dividing it at a back slash character (\) into two strings. The first being the identifier and the second being the type. In this eventuality special processing should be present to manage the two pieces of inform.	

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
The code supporting the next two steps must only be executed upon successful registration of the service with the "identifier\type" format as the input name.				
D. Verify that the *registerService* operation associates the identifier portion of the input name parameter (in the "identifier/type" format) to the reference to the service in the DomainFinder. (OE0771)				
5. Verify that the *registerService* operation associates the identifier portion of the input name parameter to the reference to the service in the DomainFinder.	**Pass:** The *registerService* operation associates the identifier portion of the input name parameter to the reference to the service in the DomainFinder. (OE0771) **Fail:** The *registerService* operation does not associate the identifier portion of the input name parameter to the reference to the service in the DomainFinder. (OE0771)			
E. Verify that the *registerService* operation associates the type portion of the input name parameter (in the "identifier/type" format) to the reference to the service in the DomainFinder. (OE0772)				

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Verify that the *registerService* operation associates the type portion of the input name parameter to the reference to the service in the DomainFinder.	**Pass:** The *registerService* operation associates the type portion of the input name parameter to the reference to the service in the DomainFinder. (OE0772) **Fail:** The *registerService* operation does not associate the type portion of the input name parameter to the reference to the service in the DomainFinder. (OE0772)			
F. Locate the source code that supports the *domainfinder* mechanism. (OE0771, OE0772)				
7. Locate the source code that supports the *domainfinder* mechanism.	**Pass**: The source code that supports the *domainfinder* mechanism is found. (OE0771, OE0772) **Untested**: The source code that supports the *domainfinder* mechanism is not found. (OE0771, OE0772)		A developer may be required to identify this location	

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
G. Verify that for a get-object-reference operation within the domainfinder mechanism using "servicename" as its input object type, the operation will search for a matching name from the set of "servicename" identifiers that has been registered with the domain. (OE0771)				
8. Verify that a get-object-reference operation within the *domainfinder* mechanism using "servicename" as its input object type, the operation will search for a matching name from the set of "servicename" identifiers that has been registered with the domain.	**Pass:** The get-object-reference operation within the *domainfinder* mechanism using "servicename" as its input object type, searches for a matching name from the set of "servicename" identifiers that has been registered with the domain. (OE0771) **Fail:** The get-object-reference operation within the *domainfinder* mechanism using "servicename" as its input object type, does not search for a matching name from the set of "servicename" identifiers that has been registered with the domain. (OE0771)			
H. Verify that for a get-object-reference operation within the domainfinder mechanism using "servicetype" as its input object type, the operation will search for a matching name from the set of "servicetype" types that has been registered with the domain. (OE0772)				

OE_TC_176				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
9. Verify that a get-object-reference operation within the *domainfinder* mechanism using "servicetype" as its input object type, the operation will search for a matching name from the set of "servicetype" types that has been registered with the domain.	**Pass:** The get-object-reference operation within the *domainfinder* mechanism using "servicetype" as its input object type, searches for a matching name from the set of "servicetype" types that has been registered with the domain. (OE0772) **Fail:** The get-object-reference operation within the *domainfinder* mechanism using "servicetype" as its input object type, does not search for a matching name from the set of "servicetype" types that has been registered with the domain. (OE0772)			
End of Test				

Test Recording Log – OE_TC_176		
Step 1 & 2 (locate and inspect DCD files)		
Step 3 (locate implementation of DomainMgr::registerService)		
Step 4 (determines when the "identifier\type" format is being used)	**Step 5** (stores the identifier portion)	**Step 6** (stores the type portion)
Step 7 (locate the source for the *domainfinder* mechanism)		
Step 8 (uses "servicename" and a name to search for object)	**Step9** (uses "servicetype" and a type to search for object)	

Test Recording Log – OE_TC_176		
Step 1 & 2 (locate and inspect DCD files)		
Step 3 (locate implementation of DomainMgr::registerService)		
Step 4 (determines when the "identifier\type" format is being used)	**Step 5** (stores the identifier portion)	**Step 6** (stores the type portion)

Test Summary OE_TC_176

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0771_____

OE0772 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.2 OE_TC_177 - DomainManager :: unregisterService removes non-SCA services

Test Case Number: OE_TC_177

DomainManager::unregisterService

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0773	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-48 , Section 3.1.3.2.3.6.8.3
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 1, Section 2.2 Page 7, Section 6.10
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-21

Test Objective

This test case verifies OE0773. The objective of this test is to verify that during the unregistering of a non-SCA service with an identifier and type in the "identifier\type" format, the *unregisterService* operation removes the service using either the fully qualified name or a simple name using the "identifier" portion only.

SCA services are those identified by the *findby* element in the SCA Appendix D. The SCA Extensions add two types to the attribute definition of the *domainfinder* element, a child of the findby element. The two types as described in section 6.10 of the SCA Extensions document are 'servicename' and 'servicetype'. Using either of these two types automatically indicate that the service is non-SCA. These requirements are saying that the unregistering process uses the type, the string that follows the first back slash of the input name, and identifier, the string that precedes the first back slash of the input name, or just the identifier within the domainfinder mechanism to retrieve the registered service for the unregistering process.

Note: The test case uses the term **mechanism** to mean a collection of software in support of the named functionality. Being more specific may lead to a programming language dependent discussion.

Definitions:

> "identifier " as used above is that portion of the input name that exists before the first back slash character in the "identifier\type" format.
>
> "type" as used above is that portion of the input name that exists after the first back slash character in the "identifier\type" format.
>
> "fully qualified name" means using both the identifier and the type from the "identifier\type" format.
>
> "simple name" means using only the identifier from the "identifier\type" format.

Places to Verify

DomainManager and the support software for the *domainfinder*.

IDL References

Operations

void unregisterService (in Object unregisteringService, in string name)

> raises (InvalidObjectReference, UnregisterError);

Preconditions

- The DomainManager source code files are available.
- The source code files, which support the domainfinder mechanism, are available.
- The SCA Extension requirements are implemented.

Test Description

For each domainManager within the OE under test, perform the following steps:

A. Locate the source code for the DomainManager *unregisterService* operation. (OE0773)
   1. **Pass**: The source code for the DomainManager *unregisterService* operation is found.

2. **Untested**: The source code for the DomainManager *unregisterService* operation is not found.

For each *unregisterService* operation found

B. Verify that the *unregisterService* operation processes the "identifier\type" format. (OE0773)
   1. **Pass:** The *unregisterService* operation processes the "identifier\type" format.
   2. **Fail:** The *unregisterService* operation does not process the "identifier\type" format.
   3. **Untested:** No implementations of the DomainManager's *unregisterService* operation can be found.
C. Verify that the *unregisterService* operation identifies the service to act on and removes it.
   1. Verify that the *unregisterService* operation identifies the service to act on either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion. (OE0773)
      a. **Pass:** The *unregisterService* operation identifies the service to unregister either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion.
      b. **Fail:** The *unregisterService* operation does not identify the service to unregister either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion.
   2. Verify that the *unregisterService* operation removes the identified service from the list of registered services. (OE0773)
      a. **Pass:** The *unregisterService* operation removes the identified service from the list of registered services.
      b. **Fail:** The *unregisterService* operation does not remove the identified service from the list of registered services.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_177				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
For each DomainManager within the OE under test				
A. Locate the source code for the DomainManager *unregisterService* operation. (OE0773)				
1. Perform a search on all source code provided by the developer for the implementation of the *DomainManager unregisterService* operation and record the source code file name.	**Pass:** The source code for the *DomainManager unregisterService* operation is found. (OE0773) **Untested**: The source code for the *DomainManager unregisterService* operation is not found. (OE0773)			
For each *unregisterService* operation found				
B. Verify that the *unregisterService* operation processes the "identifier\type" format. (OE0773)				

OE_TC_177				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Verify that the *unregisterService* operation processes the "identifier\type" format that is being used in the input name parameter.	**Pass:** The *unregisterService* operation processes the "identifier\type" format that is being used in the input name parameter. (OE0773) **Fail:** The *unregisterService* operation does not process the "identifier\type" format that is being used in the input name parameter. (OE0773)		The code should attempt to parse the input name parameter, dividing it at a back slash character (\) into two strings. The first part being the identifier and the second part being the type. The *unregisterService* operation should then use these values.	
C. Verify that the *unregisterService* operation identifies the service to act on and removes it.				
1. Verify that the *unregisterService* operation identifies the service to act on either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion. (OE0773)				

OE_TC_177				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the *unregisterService* operation identifies the service to unregister either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion to an entry in a list of services.	**Pass:** The *unregisterService* operation identifies the service to unregister either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion to an entry in a list of services. (OE0773) **Fail:** The *unregisterService* operation does not identify the service to unregister either by matching the fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion to an entry in a list of services. (OE0773)		A fully qualified name means using both portions (identifier and type) of the parsed parameter from step 2. A simple name means that the processing only uses the identifier portion of the parsed string.	
2. Verify that the *unregisterService* operation removes the identified service from the list of registered services. (OE0773)				

OE_TC_177				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that the unregisterService operation removes the identified service from the list of registered services.	**Pass:** The *unregisterService* operation removes the identified service from the list of registered services. (OE0773) **Fail:** The *unregisterService* operation does not remove the identified service from the list of registered services. (OE0773)			
End of Test				

Test Recording Log – OE_TC_177		
Step 1 (locate implementation of DomainMgr::unregisterService)	**Step 2** (determines if the "identifier\type" format is being processed)	**Step 3 & 4** (uses "identifier\type" or "identifier" to find and remove service)

Test Summary OE_TC_177

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass: No failures detected

Fail: Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.

Untested: Condition which is not testable

N/A: Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0773_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.3 OE_TC_178 - ApplicationFactory :: create recognizes DEPLOYMENT_CHANNEL options

Test Case Number: OE_TC_178

ApplicationFactory::create

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0774	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.
OE0775	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.
OE0776	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.
OE0777	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.
OE0778	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Pages 3-27 thru 3-31, Section 3.1.3.2.2.5.1.3
SCA Extentions	Version 2.2.2 22 December 2006	Pages 1 thru 2, Section 2.3 Page 3, Section 2.8 and 2.9 Pages 9 thru 10, Section 6.12 & 6.13
SCA Appendix C Core Framework IDL	Version 2.2.2 FINAL/15 May 2006	Page C-23

Test Objective

This test case verifies OE0774, OE0775, OE0776, OE0777 and OE0778. The objective of this test is to verify that applications are deployed per the channel preferences based on the information provided in the Application Deployment Descriptor (ADD) file, the Deployment Platform Descriptor (PDD) file and the DEPLOYMENT_CHANNEL property in the *create* operation parameter list.

Definitions:

"application deployment channel preferences" from the ADD file is described by the XML element called *deploymentprecedence* and is detailed in the IDL Reference section below. From the PDD file, it is describe by the XML element called *deploymentplatform*. Details of this may be found below as well. Finally, there is a new property called the DEPLOYMENT_CHANNEL property, which identifies the preferences of application deployment.

The idea is that the user may direct the deployment of applications during the initialization process. These various methods give the user flexibility to do so.

Requirement OE0777 verifies the allocation of an application in accordance to the DPP or the ADD. Since the ADD is an application profile referenced in the SAD, it is not expected that the target OE product will have this referenced file available for evaluation. The following procedures assume that the SAD properly references an ADD and the ADD exists in the application.

Places to Verify

ApplicationFactory source code and the Domain Profile files.

IDL References

Operations

```
CF::Application create (
                    in string name,
                    in CF::Properties initConfiguration,
                    in CF::DeviceAssignmentSequence deviceAssignments)
        raises ( CF::ApplicationFactory::CreateApplicationError,
                CF::ApplicationFactory::CreateApplicationRequestError,
```

CF::ApplicationFactory::InvalidInitConfiguration);

SCA Extensions ADD XML

<!ELEMENT deploymentprecedence

(description?

, deploymentoptions) >

<!ELEMENT description (#PCDATA)>

<!ELEMENT deploymentoptions (deploymentoption+)>

<!ELEMENT deploymentoption (description? , channelref+)>

<!ATTLIST deploymentoption

deployedname CDATA #REQUIRED>

<!ELEMENT channelref EMPTY>

<!ATTLIST channelref

refname CDATA #REQUIRED>

SCA Extensions PDD XML

<!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>
```

Preconditions

- The ApplicationFactory source code files are available.
- All domain profile files including *.pdd files.
- The OE requires SCA Extension testing.

Test Description

A. Locate the source code for the ApplicationFactory *create* operation. (OE0774, OE0775, OE0776, OE0777, and OE0778)
   1. **Pass**: The source code for the ApplicationFactory *create* operation is found.
   2. **Fail**: The source code for the ApplicationFactory *create* operation is not found.

B. Support for enhanced deployment via the use of a PDD file is optional.
   1. Verify that the DMD file references an existing PDD file using the *deploymentlayout* element. (OE0775, OE0776, OE0777, OE0778)
      a. **Pass:** The DMD file does reference an existing PDD file.
      b. **N/A:** The DMD file does not reference an existing PDD file.

   If the use of the PDD file is not applicable(B.1.b), then stop the test. Otherwise, perform all of the following steps:

   2. Verify that the *create* operation recognizes application deployment channel preferences. These are ones contained within an ADD file. (OE0774)

a. **Pass:** The *create* operation recognizes application deployment channel preferences, which are contained within an ADD file.
b. **Fail:** The *create* operation does not recognize application deployment channel preferences, which are contained within an ADD file.

3. Verify that the *create* operation recognizes a property which is a CF Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence. (OE0775)
   a. **Pass:** The *create* operation recognizes a property which is a CF Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence.
   b. **Fail:** The *create* operation does not recognize a property which is a CF Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence.
4. Verify that the *create* operation recognizes channel preferences contained within a "DEPLOYMENT_CHANNEL" property contained within the initConfiguration parameter. (OE0776)
   a. **Pass:** The *create* operation recognizes channel preferences contained within a "DEPLOYMENT_CHANNEL" property contained within the initConfiguration parameter.
   b. **Fail:** The *create* operation does not recognize channel preferences contained within a "DEPLOYMENT_CHANNEL" property contained within the initConfiguration parameter.
5. Verify that the *create* operation attempts to allocate an application to the PDD file channel alternatives provided within a "DEPLOYMENT_CHANNEL" property or an ADD file in a sequential manner. (OE0777)
   a. **Pass:** The *create* operation attempts to allocate an application to the PDD file channel alternatives provided within a "DEPLOYMENT_CHANNEL" property or an ADD file in a sequential manner.
   b. **Fail:** The *create* operation does not attempt to allocate an application to the PDD file channel alternatives provided within a "DEPLOYMENT_CHANNEL" property or an ADD file in a sequential manner.
6. Verify that the *create* operation utilize channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained within an ADD file. (OE0778)
   a. **Pass:** The *create* operation utilize channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained within an ADD file.
   b. **Fail:** The *create* operation does not utilize channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained within an ADD file.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for the ApplicationFactory *create* operation. (OE0774, OE0775, OE0776, OE0777, and OE0778)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned here and search for *create* operation implementation. Record the file name.	**Pass**: The source code for the ApplicationFactory *create* operation is found. Record the file name. (OE0774, OE0775, OE0776, OE0777, OE0778) **Fail**: The source code for the ApplicationFactory *create* operation is not found. (OE0774, OE0775, OE0776, OE0777, OE0778)			
B. Support for enhanced deployment via the use of a PDD file is optional.				
1. Verify that the DMD file references an existing PDD file using the *deploymentlayout* element. (OE0774, OE0775, OE0776, OE0777, OE0778)				

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Search the DMD file for the *deploymentlayout* element.	**Pass**: The *deploymentlayout* element is found. (OE0774, OE0775, OE0776, OE0777, OE0778) **N/A**: The *deploymentlayout* element is not found. (OE0774, OE0775, OE0776, OE0777, OE0778)			
If step 2 above failed, stop test. The *deploymentlayout* element must contain the reference of the PDD. See Preconditions.				
3. Verify that the *localfile* element within the *deploymentlayout* element points to an existing PDD file.	**Pass**: The *localfile* element within the *deploymentlayout* element points to an existing PDD file. Record the name of the file, (OE0774, OE0775, OE0776, OE0777, OE0778) **N/A**: The *localfile* element within the *deploymentlayout* element does not point to an existing PDD file. (OE0774, OE0775, OE0776, OE0777, OE0778)			
If step 3 above failed, stop test. The PDD file must exist in the domain profile. See Preconditions.				
2. Verify that the create operation recognizes application deployment channel preferences. These are ones contained within an ADD file. (OE0774)				

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Find within the *create* operation where application deployment channel preferences are processed.	**Pass**: The *create* operation processes application deployment channel preferences. (OE0774) **Fail**: The *create* operation does not process application deployment channel preferences. (OE0774)		We are concerned here about the create operation's processing of channel, not their definition in the ADD file. Section 6.12.1 of the SCA Extensions document says the following about the ADDs *deploymentprecedence*: (aka application deployment channel preferences) "The *deploymentprecedence* element contains the relationship between application instances and their candidate virtual channels."	
5. In the *create* operation source code where the deployment channel preferences are made, verify that the contents of the ADDs *deploymentprecedence* element is processed.	**Pass**: The ADD's *deploymentprecedence* element is processed. (OE0774) **Fail**: The ADD's *deploymentprecedence* element is not processed. (OE0774)		*The XML definition of deploymentprecedence* appears in the IDL Reference section of this test case.	

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the *create* operation recognizes a property which is a CF Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence. (OE0775)				
6. In the *create* operation source code where the deployment channel preferences are made, verify that there is processing for a CF::Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence.	**Pass**: The *create* operation source code where the deployment channel preferences are made contains processing for a CF::Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence. (OE0775) **Fail**: The *create* operation source code where the deployment channel preferences are made does not contain processing for a CF::Properties type with an id of "DEPLOYMENT_CHANNEL" and a value that is a string sequence. (OE0775)			
4. Verify that the *create* operation recognizes channel preferences contained within a "DEPLOYMENT_CHANNEL" property contained within the initConfiguration parameter. (OE0776)				

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
7. In the *create* operation source code where the input parameter initConfiguration is parsed, verify there is support for preferences for a property with the id of "DEPLOYMENT_CHANNEL".	**Pass**: The *create* operation source code where the input parameter initConfiguration is parsed, contains support for preferences for a property with an id of "DEPLOYMENT_CHANNEL". (OE0776) **Fail**: The *create* operation source code where the input parameter initConfiguration is parsed, does not contain support for preferences for a property with an id of "DEPLOYMENT_CHANNEL". (OE0776)			
5. Verify that the *create* operation attempts to allocate an application to the PDD file channel alternatives provided within a "DEPLOYMENT_CHANNEL" property or an ADD file in a sequential manner. (OE0777)				

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
8. In the *create* operation source code verify that the attempts to allocate an application, either to the PDD file or the ADD file channel alternatives, are in a sequential manner.	**Pass**: The attempts to allocate an application to the PDD file or the ADD file channel alternatives, are in a sequential manner. (OE0777) **Fail**: The attempts to allocate an application to the PDD file or the ADD file channel alternatives, are not in a sequential manner. (OE0777)		There are channel numbers identified for applications in either the ADD file or the PDD file. This requirement is stating they need to be allocated in sequential channel order.	
6. Verify that the create operation utilize channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained within an ADD file. (OE0778)				

OE_TC_178				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
9. Verify that within the processing of step 9, that the processing utilizes channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained in an ADD file.	**Pass**: The processing to allocate an application utilizes channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained in an ADD file. (OE0777) **Fail**: The processing to allocate an application does not utilize channel preferences expressed within a "DEPLOYMENT_CHANNEL" property rather than those contained in an ADD file. manner. (OE0777)		There may be channel references mentioned in the ADD file, the PDD file's "DEPLOYMENT_CHANNEL" property or a "DEPLOYMENT_CHANNEL" property in the initConfiguration input parameter. This requirement states that the "DEPLOYMENT_CHANNEL" property defined channel is preferred over the ADD file allocation.	
End of Test				

Test Recording Log – OE_TC_178					
Step 1 (source code for ApplicationFactory:: create)					
Step 2&3 (PDD filename)					
Step 4&5 (recognizes app deployment channel prefs)	**Step 6** (recognizes DEPLOYMENT_ CHANNEL property)	**Step 7** (recognizes DEPLOYMENT_ CHANNEL in input param)	**Step 8** (attempts to allocate apps to channel assignment is sequential)	**Step 9** (does DEPLOYMENT_ CHANNEL property override ADD file)	**Notes**

Test Summary OE_TC_178

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass: No failures detected

Fail: Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.

Untested: Condition which is not testable

N/A: Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0774_____

OE0775_____

OE0776 _____

OE0777 _____

OE0778 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.4 OE_TC_179 - ApplicationFactory :: create recognizes DEFAULT deployment options

Test Case Number: OE_TC_179

ApplicationFactory::create

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0779	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Pages 3-27 through 3-31, Section 3.1.3.2.2.5.1.3
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-33, Section C.1
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-33 through D-47; Section D.6.1
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 3, Section 2.9; Page 6, Section 6.8; Page 9 through 10, Section 6.12;

Test Objective

This test case verifies OE0779. The objective of this test is to verify that if the CF implementation provides enhanced deployment support via the use of an Application Deployment Descriptor (ADD) file, then the ApplicationFactory *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. That is, if the CF is using the SCA Extension ADD file, then the ApplicationFactory *create* operation will know about a default deployment option and match it with application names that are not identified with deployment options

SCA Extension specific differences:

SAD files may reference a new file (ADD) intended for defining application channel deployments preferences.

The string "DEFAULT" is reserved as a *deployedname* attribute for defining a default channel for applications not given a preferred channel.

Places to Verify

ApplicationFactory

IDL References

Operations

CF::Application create (

in string name,

in CF::Properties initConfiguration,

in CF::DeviceAssignmentSequence deviceAssignments)

raises (CF::ApplicationFactory::CreateApplicationError,

CF::ApplicationFactory::CreateApplicationRequestError,

CF::ApplicationFactory::InvalidInitConfiguration); };

Preconditions

- The ApplicationFactory source code files are available.
- The Domain Profile files are available.
- The OE requires SCA Extension testing.
- It is assumed that there exists a SAD file that supports SCA Extension testing.
- It is assumed that there exists an ADD file with a deployment option(*deploymentoption* element) containing a *deployedname* attribute value of "DEFAULT" and a valid channel value (*channelref* element).

Test Description

A. Locate the source code for the ApplicationFactory *create* operation. (OE0779)
   1. **Pass**: The source code for the ApplicationFactory *create* operation is found.

2. **Fail**: The source code for the ApplicationFactory *create* operation is not found.

For each ApplicationFactory *create* operation found within the OE under test do the following steps:

B. Verify that the create operation searches a collection of deployment options using the input application name. (OE0779)
   1. **Pass:** The *create* operation searches a collection of deployment options using the input application name.
   2. **Fail:** The *create* operation does not search a collection of deployment options using the input application name.

C. Verify that the *create* operation matches the default deployment option with all application instance names that are not explicitly identified in the collection of deployment options. (OE0779)
   1. **Pass:** The *create* operation matches the default deployment option with all application instance names that are not explicitly identified in the collection of deployment options.
   2. **Fail**: The *create* operation does not match the default deployment option with all application instance names that are not explicitly identified in the collection of deployment options.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_179				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for the ApplicationFactory *create* operation. (OE0779)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned and search for *create* operation implementation. Record the file name.	**Pass**: The source code for the ApplicationFactory *create* operation is found. Record the file name. (OE0779) **Fail**: The source code for the ApplicationFactory *create* operation is not found. (OE0779)			
For each ApplicationFactory *create* operation found within the OE under test do the following steps:				
B. Verify that the create operation searches a collection of deployment options using the input application name. (OE0779)				

OE_TC_179				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Locate the processing where the *create* operation is verifying its input application name against a list of deployment options.	**Pass**: The *create* operation processing which verifies its input name against a list of deployment options is found. (OE0779) **Fail**: The *create* operation processing which verifies its input name against a list of deployment options is not found. (OE0779)		Deployment options are an ordered list of application names and their channel assignments.	
3. Verify that there is a default deployment option value. That is, there is processing to address when an application name does not appear on the deployment options list.	**Pass**: There is a default deployment option value. (OE0779) **Fail**: There is no default deployment option value. (OE0779)			
C. Verify that the *create* operation matches the default deployment option with all application instance names that are not explicitly identified in the collection of deployment options. (OE0779)				

OE_TC_179				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that when the input name is not in the deployment options list, that the default deployment option value is used to deploy the application.	**Pass**: The input name is not in the deployment options list, and the default deployment option value is used to deploy the application. (OE0779) **Fail**: The input name is not in the deployment options list, but the default deployment option value is not used to deploy the application. (OE0779)			
End of Test				

Test Recording Log – OE_TC_179		
Step 1 (File name)	**Step 2 & 3** (*create* op searches deployment list) Y/N	**Step 4** (*create* op uses default value when search fails) Y/N

Test Summary OE_TC_179

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass: No failures detected

Fail: Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.

Untested: Condition which is not testable

N/A: Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0779_____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.5 OE_TC_180 - ApplicationFactory :: create with "servicetype" connections to a non-SCA service

Test Case Number: OE_TC_180

ApplicationFactory::create with "servicetype" connections to a non-SCA service

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0780	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Pages 3-27 through 3-31, Section 3.1.3.2.2.5.1.3
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 2, Section 2.3; Pages 8 through 9, Section 6.11; Page 11, Section 6.13;
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-23
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-54 through D-55; Section D.8.1

Test Objective

This test case verifies OE0780. The objective of this test is to verify that if the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor (PDD) file, 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. That is, if the CF is using the SCA Extension PDD file, then for "servicetype" connections whose service type is provided via a channel element servicelist, the *create* operation will only attempt to establish connections to the services within the list. Note: "non-SCA services" are those other than Log, FileSystem, Event and Naming services.

Places to Verify

ApplicationFactory

IDL References

Operations

CF::Application create (

in string name,

in CF::Properties initConfiguration,

in CF::DeviceAssignmentSequence deviceAssignments)

raises (CF::ApplicationFactory::CreateApplicationError,

CF::ApplicationFactory::CreateApplicationRequestError,

CF::ApplicationFactory::InvalidInitConfiguration); };

Preconditions

- The ApplicationFactory source code files are available.
- The Domain Profile files are available.
- The OE requires SCA Extension testing.
- A DMD file that supports SCA Extension testing. That is the DMD file references an existing PDD file using the *deploymentlayout* element.

Test Description

A. Locate the source code for the ApplicationFactory *create* operation. (OE0780)
   1. **Pass**: The source code for the ApplicationFactory *create* operation is found.
   2. **Fail**: The source code for the ApplicationFactory *create* operation is not found.

For each ApplicationFactory *create* operation found:

B. Verify that the create operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist". (OE0780)

1. **Pass:** The *create* operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist".
2. **Fail**: The *create* operation does not process "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist".

C. Verify that the *create* operation only attempts to establish connections to services within the *channel* element servicelist. (OE0780)
1. **Pass:** The *create* operation only attempts to establish connections to services within the *channel* element servicelist.
2. **Fail**: The *create* operation does not attempt to establish connections to services within the *channel* element servicelist.
3. **Fail**: The *create* operation attempts to establish connections to services not within the *channel* element servicelist.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_180				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for the ApplicationFactory *create* operation. (OE0780)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned in the above step and search for *create* operation implementation. Record the file name and line number.	**Pass**: The source code for the ApplicationFactory *create* operation is found. Record the file name and line number. (OE0780) **Fail**: The source code for the ApplicationFactory *create* operation is not found. (OE0780)		May need the help of the software engineer to locate the source code files directories.	
For each ApplicationFactory *create* operation found:				
B. Verify that the create operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist". (OE0780)				

OE_TC_180				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Locate in the *create* operation where the servicetype connections to non-SCA services are processed. This may be part of processing that "responds" to specific conditions within the DMD file. That specific condition in this case is the "servicetype" condition.	**Pass**: The location in the *create* operation, where the servicetype connections to non-SCA services are processed, is found. (OE0780) **Fail**: The location in the *create* operation, where the servicetype connections to non-SCA services are processed, is not found. (OE0780)		The use of the "servicetype" attribute by default makes it a connection to a non-SCA service.	
3. Locate in the *create* operation where the servicetype connections to non-SCA services are narrowed to those whose service type is provided by a service contained within a *channel* element "servicelist"	**Pass**: The location in the *create* operation, where the servicetype connections are narrowed, is found. (OE0780) **Fail**: The location in the *create* operation, where the servicetype connections are narrowed, is not found. (OE0780)			
C. Verify that the *create* operation only attempts to establish connections to services within the *channel* element servicelist. (OE0780)				

OE_TC_180				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that the *create* operation only attempts to establish connections to services within the *channel* element servicelist.	**Pass**: The *create* operation only attempts to establish connections to services within the *channel* element servicelist. (OE0780) **Fail**: The *create* operation does not attempt to establish connections to services within the *channel* element servicelist. (OE0780) **Fail**: The *create* operation attempts to establish connections to services not within the *channel* element servicelist. (OE0780)			
End of Test				

Test Recording Log – OE_TC_180			
Step 1 (source file name)	**Step 2** (servicetype connects are recognized) Y/N	**Step 3** (servicetype connections are narrowed to those on servicelist) Y/N	**Step 4** (*create* op only attempts connections to such services) Y/N

Test Summary OE_TC_180

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0780 _____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.6 OE_TC_181 - ApplicationFactory :: create raises InvalidInitConfiguration

Test Case Number: OE_TC_181

ApplicationFactory::create raises InvalidInitConfiguration

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0781	The *create* operation shall raise the InvalidInitConfiguration exception when the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property contains an invalid channel reference.
OE0782	The InvalidInitConfiguration invalidProperties parameter shall identify the invalid channels.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-31, Section 3.1.3.2.2.5.1.5,
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 2, Section 2.4
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-2, C-22 and C-23

Test Objective

This test case verifies OE0781, and OE0782. The objective of this test is to verify that the ApplicationFactory *create* operation raises the *InvalidInitConfiguration* exception when the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property contains an invalid channel reference. Furthermore, verify the exception identifies the invalid channel reference within the exception's invalidProperties parameter.

Places to Verify

ApplicationFactory

IDL References

Data

```
struct DataType {       string id;
                        any value;};

typedef sequence <DataType> Properties;
```

Exceptions

```
exception InvalidInitConfiguration { CF::Properties invalidProperties; };
```

Operations

```
CF::Application create (
                in string name,
                in CF::Properties initConfiguration,
                in CF::DeviceAssignmentSequence deviceAssignments )
        raises (CF::ApplicationFactory::CreateApplicationError,
                CF::ApplicationFactory::CreateApplicationRequestError,
                CF::ApplicationFactory::InvalidInitConfiguration);  };
```

Preconditions

- The ApplicationFactory source code files are available.
- The OE requires SCA Extension testing.

Test Description

A. Identify the source code files that implement the ApplicationFactory *create* operation. (OE0781, OE0782)
   1. **Pass**: The ApplicationFactory *create* operation source code files are found.
   2. **Fail**: The ApplicationFactory *create* operation source code files are not found.

For each ApplicationFactory *create* operation found within the OE under test do the following steps

B. Verify that the ApplicationFactory *create* operation raises the *InvalidInitConfiguration* exception when the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property contains an invalid channel reference. (OE0781).
   1. **Pass:** The *create* operation raises the *InvalidInitConfiguration* exception.
   2. **Fail:** The *create* operation does not raise the *InvalidInitConfiguration* exception.

C. Verify that the exception's invalidProperties parameter identifies the invalid channel reference. (OE0782)
   1. **Pass:** The exception's invalidProperties parameter identifies the invalid channel reference.
   2. **Fail:** The exception's invalidProperties parameter fails to identify the invalid channel reference.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_181				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement the ApplicationFactory *create* operation. (OE0781, OE0782)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned here and search for *create* operation implementations. Record the file name.	**Pass**: The implementation of the *create* operation is identified and recorded for each ApplicationFactory implementation. The file names are recorded. (OE0781, OE0782) **Fail:** Some ApplicationFactory implementation fails to have an implementation of a *create* operation. (OE0781, OE0782)		May need the help of the software engineer to locate the source code files directories.	
For each ApplicationFactory *create* operation found within the OE under test do the following steps				
B. Verify that the ApplicationFactory *create* operation raises the *InvalidInitConfiguration* exception when the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property contains an invalid channel reference. (OE0781)				
2. Search within the *create* operation for where the input initConfiguration parameter "DEPLOYMENT_ CHANNEL" property is being validated.	Identified the source code where create operation is validating the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property.			

OE_TC_181				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that when the input initConfiguration parameter "DEPLOYMENT_ CHANNEL" property has an invalid channel reference, the *InvalidInitConfiguration* exception is raised.	**Pass:** The *create* operation raises the *InvalidInitConfiguration* exception. (OE0781) **Fail:** The *create* operation fails to raise the *InvalidInitConfiguration* exception. (OE0781)			
C. Verify that the exception's invalidProperties parameter identifies the invalid channel reference. (OE0782)				
4. Verify that within the *InvalidInitConfiguration* exception processing that the invalid channel reference is stored in the invalidProperties parameter.	**Pass:** The invalid channel reference is stored in the invalidProperties parameter. (OE0782) **Fail:** The invalid channel reference is not stored in the invalidProperties parameter. (OE0782) **Fail:** There is no source code to process the *InvalidInitConfiguration* exception. (OE0782)			
End of Test				

Test Recording Log – OE_TC_181		
Step 1 (Source file name)	Step 3 (Exception raised) Y/N	Step 4 (Invalid channel reference stored in parameter) Y/N

Test Summary OE_TC_181

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0781_____

OE0782_____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.7 OE_TC_182 - ApplicationFactory :: create raises CreateApplicationError when not able to allocate applications properly

Test Case Number: OE_TC_182

ApplicationFactory::create raises CreateApplicationError

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 and SCA Extensions v2.2.2 Text
OE0783	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.
OE0152	The error number shall indicate a CF ErrorNumberType value.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-26, Section 3.1.3.2.2.3.2, Page 3-94, Section 3.1.3.6.13
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 2, Section 2.4
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-3 to C-4, C-23

Test Objective

This test case verifies OE0783 and OE0152. The objective of this test is to verify that the ApplicationFactory *create* operation raises the *CreateApplicationError* exception when the CF implementation provides enhanced deployment support via the use of a Deployment Platform Descriptor (PDD) file but the CF is not able to allocate the application to any of the provided channel alternatives. That is, if the CF is using the SCA Extensions defined PDD file, then the ApplicationFactory *create* operation should raise the *CreateApplicationError* exception, if the CF is not able to allocate the application to any of the provided channel alternatives.

Notes: "provided channel alternatives" is a reference to section 2.3 of the SCA Extensions document. This section describes several methods for the new PDD and Application Deployment Descriptor (ADD) files to provide deployment channel information to the application. This discussion is represented by new SCA Extension requirements, OE0774, OE0775, OE0776, OE0777, and OE0778. These requirements are tested in OE_TC_178.

Places to Verify

ApplicationFactory

IDL References

Data

enum ErrorNumberType {

CF_NOTSET, CF_E2BIG, CF_EACCES, CF_EAGAIN, CF_EBADF, CF_EBADMSG, CF_EBUSY, CF_ECANCELED, CF_ECHILD, CF_EDEADLK, CF_EDOM, CF_EEXIST, CF_EFAULT, CF_EFBIG, CF_EINPROGRESS, CF_EINTR, CF_EINVAL, CF_EIO, CF_EISDIR, CF_EMFILE, CF_EMLINK, CF_EMSGSIZE, CF_ENAMETOOLONG, CF_ENFILE, CF_ENODEV, CF_ENOENT, CF_ENOEXEC, CF_ENOLCK, CF_ENOMEM, CF_ENOSPC, CF_ENOSYS, CF_ENOTDIR, CF_ENOTEMPTY, CF_ENOTSUP, CF_ENOTTY, CF_ENXIO, CF_EPERM, CF_EPIPE, CF_ERANGE, CF_EROFS, CF_ESPIPE, CF_ESRCH, CF_ETIMEDOUT, CF_EXDEV };

Exceptions

exception CreateApplicationError { CF::ErrorNumberType errorNumber; string msg; };

Operations

```
CF::Application  create (
                in string name,
                in CF::Properties initConfiguration,
                in CF::DeviceAssignmentSequence deviceAssignments )
        raises (CF::ApplicationFactory::CreateApplicationError,
                CF::ApplicationFactory::CreateApplicationRequestError,
                CF::ApplicationFactory::InvalidInitConfiguration);  };
```

Preconditions

- The ApplicationFactory source code files are available.
- The Domain Profile files are available.
- The OE requires SCA Extension testing.
- A DMD file that supports SCA Extension testing. That is the DMD file references an existing PDD file using the *deploymentlayout* element.

Test Description

A. Locate the source code for the ApplicationFactory *create* operation. (OE0783)

1. **Pass**: The source code for the ApplicationFactory *create* operation is found.
2. **Fail**: The ApplicationFactory *create* operation is not found.

For each ApplicationFactory *create* operation found within the OE under test:

B. Verify that the ApplicationFactory *create* operation raises the CF::*CreateApplicationError* when the CF is not able to allocate the application to any of the provided channel alternatives. (OE0783).

1. **Pass:** The *create* operation raises the *CreateApplicationError* exception when the CF is not able to allocate the application to any of the provided channel alternatives.
2. **Fail:** The create operation does not raise the CreateApplicationError exception when the CF is not able to allocate the application to any of the provided channel alternatives.

C. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0152)

1. **Pass:** The *CreateApplicationError* exception provides an error number with a CF ErrorNumberType value for the error condition.
2. **Fail:** The CreateApplicationError exception does not match the error number provided in CF ErrorNumberType, nor does it correctly match the error condition.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_182				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for the ApplicationFactory *create* operation. (OE0783)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned in the above task and search for *create* operation implementation. Record the file names and line numbers.	**Pass**: The source code for the ApplicationFactory *create* operation is found. Record the file names. (OE0783) **Fail**: The source code for the ApplicationFactory *create* operation is not found. (OE0783)		May need the help of the software engineer to locate the source code files directories.	
For each ApplicationFactory *create* operation found within the OE under test:				
B. Verify that the ApplicationFactory *create* operation raises the CF::*CreateApplicationError* when the CF is not able to allocate the application to any of the provided channel alternatives. (OE0783).				
2. Locate where the *create* operation attempts to allocate an application to a deployment channel.	**Pass**: The source code where the *create* operation attempts to allocate an application to a deployment channel is found. (OE0783) **Fail**: The source code where the *create* operation attempts to allocate an application to a deployment channel is not found. (OE0783)			

OE_TC_182				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Locate where the *create* operation addresses a failure to allocate an application on a deployment channel.	**Pass**: The source code where the *create* operation addresses a failure to allocate an application to a deployment channel is found. (OE0783) **Fail**: The source code where the *create* operation addresses a failure to allocate an application to a deployment channel is not found. (OE0783)			
4. Verify that the *CreateApplicationError* exception is raised when the conditions in the above step are met.	**Pass:** The *create* operation raises the CF::*CreateApplicationError* exception. (OE0783) **Fail:** The *create* operation does not raise the CF::*CreateApplicationError*. (OE0783)			
C. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0152)				
5. Determine if the exception includes an error number (errorNumber) of the type ErrorNumberType.	**Pass:** The exception includes an error number (errorNumber) of ErrorNumberType. (OE0152) **Fail:** The exception does not include an error number (errorNumber) of ErrorNumberType. (OE0152)			
End of Test				

Test Recording Log – OE_TC_182				
Step 1 (source file name)	**Steps 2 & 3** (Found where apps not allocated to provided channel alternatives) Y/N	**Step 4** (Exception raised) Y/N	**Step 5** (Error message present) Y/N	**Notes**

Test Summary OE_TC_182

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0783_____

OE0152 _____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.8 OE_TC_183 - ApplicationFactory :: create raises CreateApplicationError when a "servicetype" connection cannot be established

Test Case Number: OE_TC_183

ApplicationFactory::create raises CreateApplicationError with “servicetype” connections to a non-SCA service

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 and SCA Extensions v2.2.2 Text
OE0784	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.
OE0152	The error number shall indicate a CF ErrorNumberType value.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-26, Section 3.1.3.2.2.3.2, Page 3-94, Section 3.1.3.6.13
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 2, Section 2.4 Pages 8 through 9, Section 6.11; Page 11, Section 6.13;
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-3 to C-4, C-23
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-54 through D-55; Section D.8.1

Test Objective

This test case verifies OE0784 and OE0152. The objective of this test is to verify that the ApplicationFactory *create* operation raises the *CreateApplicationError* exception when the CF implementation provides enhanced deployment support via the use of a Deployment

Platform Descriptor (PDD) 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 cannot be established to a service identified within that list. That is, if the CF is using the SCA Extensions defined PDD file and a connection whose service type is provided by a service contained within a channel element servicelist, then the ApplicationFactory *create* operation should raise *CreateApplicationError* exception, if the a domainfinder element "servicetype" connection to a non-SCA service cannot be established to a service identified within that list.

Places to Verify

ApplicationFactory

IDL References

Data

enum ErrorNumberType {

CF_NOTSET, CF_E2BIG, CF_EACCES, CF_EAGAIN, CF_EBADF, CF_EBADMSG, CF_EBUSY, CF_ECANCELED, CF_ECHILD, CF_EDEADLK, CF_EDOM, CF_EEXIST, CF_EFAULT, CF_EFBIG, CF_EINPROGRESS, CF_EINTR, CF_EINVAL, CF_EIO, CF_EISDIR, CF_EMFILE, CF_EMLINK, CF_EMSGSIZE, CF_ENAMETOOLONG, CF_ENFILE, CF_ENODEV, CF_ENOENT, CF_ENOEXEC, CF_ENOLCK, CF_ENOMEM, CF_ENOSPC, CF_ENOSYS, CF_ENOTDIR, CF_ENOTEMPTY, CF_ENOTSUP, CF_ENOTTY, CF_ENXIO, CF_EPERM, CF_EPIPE, CF_ERANGE, CF_EROFS, CF_ESPIPE, CF_ESRCH, CF_ETIMEDOUT, CF_EXDEV };

Exceptions

exception CreateApplicationError { CF::ErrorNumberType errorNumber; string msg; };

Operations

CF::Application create (

in string name,

in CF::Properties initConfiguration,

in CF::DeviceAssignmentSequence deviceAssignments)

raises (CF::ApplicationFactory::CreateApplicationError,

CF::ApplicationFactory::CreateApplicationRequestError,

CF::ApplicationFactory::InvalidInitConfiguration); };

Preconditions

- The ApplicationFactory source code files are available.
- The Domain Profile files are available.
- The OE requires SCA Extension testing.
- A DMD file that supports SCA Extension testing. That is the DMD file references an existing PDD file using the *deploymentlayout* element.

Test Description

A. Locate the source code for the ApplicationFactory *create* operation. (OE0784)
   1. **Pass**: The source code for the ApplicationFactory *create* operation is found.
   2. **Failed**: The source code for the ApplicationFactory *create* operation is not found.

For each ApplicationFactory *create* operation found:

B. Verify that the *create* operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist". (OE0784)
   1. **Pass:** The *create* operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist".
   2. **Fail**: The *create* operation does not process "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist".

C. Verify that the ApplicationFactory *create* operation raises the *CreateApplicationError* exception when the CF cannot establish "servicetype" connections to a service identified within that list. (OE0784)
   1. **Pass:** The *create* operation raises the *CreateApplicationError* exception when the CF is not able to allocate the application to any of the provided channel alternatives.
   4. **Fail:** The *create* operation does not raise the *CreateApplicationError* exception when the CF is not able to allocate the application to any of the provided channel alternatives.

D. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0152)
   1. **Pass:** The *CreateApplicationError* exception provides an error number with a CF ErrorNumberType value for the error condition.

2. **Fail:** The *CreateApplicationError* exception does not provide an error number with a CF ErrorNumberType value for the error condition.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_183				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for the ApplicationFactory *create* operation. (OE0784)				
1. Perform a keyword search for *create* operation on all ApplicationFactory source code provided by the developer. Examine the source code files returned here and search for *create* operation implementations. Record the file name.	**Pass**: The source code for the ApplicationFactory *create* operation is found. Record the file name. (OE0784) **Fail**: The source code for the ApplicationFactory *create* operation is not found. (OE0784)		May need the help of the software engineer to locate the source code files directories.	
For each ApplicationFactory *create* operation found:				
B. Verify that the create operation processes "servicetype" connections to a non-SCA service whose service type is provided by a service contained within a *channel* element "servicelist". (OE0784)				

OE_TC_183				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Locate in the *create* operation where the servicetype connections to non-SCA services are processed.	**Pass**: The location in the *create* operation, where the servicetype connections to non-SCA services are processed, is found. (OE0784) **Fail**: The location in the *create* operation, where the servicetype connections to non-SCA services are processed, is not found. (OE0784)		The use of the "servicetype" attribute by default makes it a connection to a non-SCA service.	
3. Locate in the *create* operation where the servicetype connections to non-SCA services are narrowed to those whose service type is provided by a service contained within a *channel* element "servicelist"	**Pass**: The location in the *create* operation, where the servicetype connections are narrowed, is found. (OE0784) **Fail**: The location in the *create* operation, where the servicetype connections are narrowed, is not found. (OE0784)			
C. Verify that the ApplicationFactory *create* operation raises the *CreateApplicationError* exception when the CF cannot establish "servicetype" connections to a service identified within that list. (OE0784)				

OE_TC_183				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate in the *create* operation where the narrowed servicetype connections cannot be established.	**Pass**: The location in the *create* operation, where the narrowed servicetype connections cannot be established, is found. (OE0784) **Fail**: The location in the *create* operation, where the servicetype connections cannot be established, is not found. (OE0784)			
5. Verify that in the *create* operation the *CreateApplicationError* exception is raised when the narrowed servicetype connections cannot be established.	**Pass:** The *create* operation raises the *CreateApplicationError* exception when the narrowed servicetype connections cannot be established. (OE0784) **Fail:** The *create* operation does not raise the *CreateApplicationError* exception when the narrowed servicetype connections cannot be established. (OE0784)			
D. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0152)				

OE_TC_183				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Determine if the exception includes an error number (errorNumber) of the type ErrorNumberType.	**Pass:** The exception includes an error number (errorNumber) of ErrorNumberType. (OE0152) **Fail:** The exception does not include an error number (errorNumber) of ErrorNumberType. (OE0152)			
End of Test				

Test Recording Log – OE_TC_183					
Step 1 (source file name)	**Step 2** (servicetype connects are processed) Y/N	**Step 3** (servicetype connections are narrowed to those on servicelist) Y/N	**Step 4 & 5** (*create* op raises exception if connections cannot be established) Y/N	**Step 6** (Error message present) Y/N	**Notes**

Test Summary OE_TC_183

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0784_____

OE0152 _____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.9 OE_TC_184 - DeviceManager's execparam properties

Test Case Number: OE_TC_184

DeviceManager

Requirements

SCA v2.2.2 Tag	SCA Extensions v2.2.2 Text
OE0785	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.
OE0786	The device manager shall pass the *componentinstantiation* element "execparam" properties that have values as parameters.
OE0787	The device manager shall pass "execparam" parameters' IDs and values as string values.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-53, Section 3.1.3.2.4.5
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Page 2 thru 3, Section 2.5
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-1 through C-2, Section C.1

Test Objective

This test case verifies OE0785, OE0786, and OE0787. The objective of this test is to verify that each DeviceManager correctly provides the DeviceManager IOR, the service name and *execparam* properties as parameters to the *execute* operation, when deploying non-SCA services. The DeviceManager IOR should consist of an ID of "DEVICE_MGR_IOR" and a value that is a string that is the *DeviceManager* stringified IOR. The service name should consist of an ID of "SERVICE_NAME" and a value that 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. The *execparam* should contain the *componentinstantiation* element "execparam" properties that have values as parameters. The *execparam* should contain IDs and values of type string.

Places to Verify

DeviceManager

IDL References

Data

```
struct DataType {string id;
                 any value; };

typedef sequence <DataType> Properties;
```

Operations

```
CF::ExecutableDevice::ProcessID_Type  execute (in string name,
                                               in CF::Properties  options,
                                               in CF::Properties  parameters )
raises (CF::Device::InvalidState,
        CF::ExecutableDevice::InvalidFunction,
        CF::ExecutableDevice::InvalidParameters,
        CF::ExecutableDevice::InvalidOptions,
        CF::InvalidFileName,
        CF::ExecutableDevice::ExecuteFail );
```

Preconditions

- The OE source code that contains the ExecutableDevice::*execute* operation is available.
- The Domain Profile files that contain the DeviceManager and execparam are available.
- The Domain Profile test, OE_TC_119, has passed.
- The OE requires SCA Extension testing.

Test Description

For each DeviceManager under test:

A. Determine if a non-SCA service can be deployed by the DeviceManager. (OE0785, OE0786 & OE0787)
   1. **Pass**: The DeviceManager is capable of deploying non-SCA services.
   2. **N/A**: The DeviceManager is not capable of deploying non-SCA services.

B. Verify that the DeviceManager supplies the *execute* operation the DeviceManager IOR and that it is composed properly for non-SCA services. (OE0785)
   1. **Pass**: The DeviceManager supplies the *execute* operation the DeviceManager IOR and it is composed properly for non-SCA services.
   2. **Fail**: The DeviceManager does not supply the *execute* operation the DeviceManager IOR or it is not composed properly for non-SCA services.

C. Verify that the DeviceManager supplies the *execute* operation the service name and it is composed properly for non-SCA services. (OE0785)
   1. **Pass**: The DeviceManager supplies the *execute* operation the service name and it is composed properly for non-SCA services.
   2. **Fail**: The DeviceManager does not supply the *execute* operation the service name or it is not composed properly for non-SCA services.

D. Identify where the *componentinstantiation* element is parsed from the XML, and verify that this is supplied as *execparam* properties. (OE0785, OE0786)
   1. **Pass**: The componentinstantiation element is supplied to the *execute* operation as *execparam* properties.
   2. **Fail**: The componentinstantiation element is not supplied to the *execute* operation as *execparam* properties.

E. Verify that the DeviceManager passes the *execparam* IDs and values as strings. (OE0785, OE0787)
   1. **Pass**: The DeviceManager passes the *execparam* IDs and values as strings.
   2. **Fail**: The DeviceManager does not pass the *execparam* IDs and values as strings.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
For each DeviceManager under test:				
A. Determine if a non-SCA service can be deployed by the DeviceManager. (OE0785, OE0786 & OE0787)				
1. Search in the source code for where the DeviceManager prepares to deploy non-SCA services.	**Pass**: The DeviceManager source code prepares to deploy non-SCA services. (OE0785, OE0786 & OE0787) **N/A**: The DeviceManager does not prepare to deploy non-SCA services. (OE0785, OE0786 & OE0787)		May need Developer support to verify where DeviceManager prepares to deploy non-SCA services. There may be multiple DeviceManager files (red, black, etc.).	

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Verify that the *execparam*, which is passed to the *execute* operation in the *parameters* argument, is of type CF::Properties.	**Pass**: The *parameters* argument that is passed to *execute* operation is of type CF::Properties. (OE0785, OE0786 & OE0787) **Fail**: The *parameters* argument that is passed to execute operation is not of type CF::Properties. (OE0785, OE0786 & OE0787)			
B. Verify that the DeviceManager supplies the *execute* operation the DeviceManager IOR and that it is composed properly for non-SCA services. (OE0785)				

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that one of the items of the *execparam* parameter consists of the id equal to "DEVICE_MGR_IOR" and the value equal to a string that is the DeviceManager's stringified IOR.	**Pass**: An item exists in the *execparam* parameter consisting of the id equal to "DEVICE_MGR_IOR" and the value equal to a string that is the DeviceManager stringified IOR (OE0785) **Fail**: No item exists in the *execparam* parameter consisting of the id equal to "DEVICE_MGR_IOR" and the value equal to a string that is the DeviceManager stringified IOR (OE0785)			
C. Verify that the DeviceManager supplies the *execute* operation the service name and it is composed properly for non-SCA services. (OE0785)				

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that one of the items of the *execparam* parameter consists of the id equal to "SERVICE_NAME" and the value is a string in an "identifier\type" format.	**Pass**: One of the items of the *execparam* parameter consists of the id equal to "SERVICE_NAME" and the value is a string in an "identifier\type" format. (OE0785) **Fail**: One of the items of the *execparam* parameter does not consist of the id equal to "SERVICE_NAME" and the value is a string in an "identifier\type" format. (OE0785)			
5. Verify, for this service name value parameter in an "identifier\type" format that the identifier portion corresponds to the DCD *componentinstantiation usagename* element.	**Pass**: The DeviceManager passes the *execparam* IDs and values as strings to the *execute* operation. (OE0785) **Fail**: The DeviceManager does not pass the *execparam* IDs and values as strings to the *execute* operation. (OE0785)		This XML component, DCD *componentinstantiation usagename* element, discussion may be found in the SCA v2.2.2 Appendix D, Domain Profile, page D-51, section D.7.1.4.1.6	

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Verify, for this service name value parameter in an "identifier\type" format that the type corresponds to a service type repository identifier from the SCD.	**Pass**: The DeviceManager passes the *execparam* IDs and values as strings to the *execute* operation. (OE0785) **Fail**: The DeviceManager does not pass the *execparam* IDs and values as strings to the *execute* operation. (OE0785)		First, find in an SCD the *componenttype* element with a value of "service". Then find the *interface* element, related to the *componenttype* element, with a *repid* attribute. That *repid* identifier is the type here	
D. Identify where the componentinstantiation element is parsed from the XML, and verify that this is supplied as *execparam* properties. (OE0785, OE0786)				
7. Verify that the *componentinstantiation element* is supplied to the *execute* operation as *execparam* properties.	**Pass**: The componentinstantiation element is supplied to the execute operation as *execparam* properties. (OE0785, OE0786) **Fail**: The componentinstantiation element is not supplied to the execute operation as *execparam* properties. (OE0785, OE0786)		This step is to ensure that the parameter contains data parsed from the XML.	
E. Verify that the DeviceManager passes the *execparam* IDs and values as strings. (OE0785, OE0787)				

OE_TC_184				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
8. Verify that the *execparam*, which is passed to the *execute* operation as a parameter, contains IDs and values as strings.	**Pass**: The DeviceManager passes the *execparam* IDs and values as strings to the *execute* operation. (OE0785, OE0787) **Fail**: The DeviceManager does not pass the *execparam* IDs and values as strings to the *execute* operation. (OE0785, OE0787)			
End of Test				

Test Recording Log – OE_TC_184						
Step 1 (DeviceManager can deploy non-SCA services – Y/N)	**Step 2** (The *execparam* is of type CF::Properties – Y/N)	**Step 3** (id of "DEVICE_MGR_IOR" and the value is string – Y/N)	**Step 4** (id of "SERVICE_NAME" and the value is "id\type" format – Y/N)	**Step 5 / 6** (id from DCD file and type from SCD file - Y/N)	**Step 7** (The *execparam* is componentinstantiation element – Y/N)	**Step 8** (The *execparam* contains IDs and values as strings – Y/N)

Test Summary OE_TC_184

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.
Untested:	Condition which is not testable
N/A:	Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0785_____

OE0786_____

OE0787_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

D.10 OE_TC_185 - Domain Profile

Test Case Number: OE_TC_185

DomainProfile for SCA Extensions

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0588	Domain Profile files shall be complaint to the Document Type Definitions (DTDs) provided in Appendix D.
OE0590	All XML files shall have as the first two lines as an XML declaration (?xml) and a document type declaration (!DOCTYPE).
OE0591	A *Software Package Descriptor* file shall have a "*.spd.xml*" extension.
OE0592	A *Software Component Descriptor* file shall have a "*.scd.xml*" extension.
OE0594	A *Properties* File shall have a "*.prf.xml*" extension.
OE0595	A *Device Package Descriptor* File shall have a "*.dpd.xml*" extension.
OE0596	A *Device Configuration Descriptor* file shall have a "*.dcd.xml*" extension.
OE0597	A *DomainManager Configuration Descriptor* file shall have a "*.dmd.xml*" extension.
OE0788	A *Deployment Platform Descriptor* file shall have a ".pdd.xml" extension.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA) Specification	Version 2.2.2 15 May 2006	Page 3-90, Section 3.1.3.5
Software Communications Architecture (SCA) Extensions	Version 2.2.2 22 December 2006	Pages 4 to 11, Section 6
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	
JTNC SCA Standards website	https://www.public.navy.mil/jtnc/Pages/resources.aspx?filter=cat-sca	

Test Objective

This test case verifies the requirements OE0588, OE0590, OE0591, OE0592, OE0594, OE0595, OE0596, and OE0597, and the extension requirement OE0788. The objective of this test is to verify that the Domain Profile is compliant to the Document Type Definitions (DTD) provided in the SCA Extensions document. The test also verifies that the Domain Profile files have the correct file extensions and that the first two lines of each Domain Profile file has the proper declaration.

Note: This test case should replace OE_TC_070 as the Domain Profile test case when testing an SCA Extension compliant OE.

Places to Verify

Domain Manager XML (Domain Profile)

IDL References

none

Preconditions

- All the Domain Profile files are available
- The OE requires SCA Extension testing.

Test Description

The following XML file name extensions, and DTD files will be verified.

Domain Profile Types, Applicable SCA DTDs and XML header line examples		
Doman Profile File Name	**File Name Extension**	**DTD File**
?xml and DOCTYPE line samples		
Deployment Platform Descriptor (PDD)	pdd.xml	deploymentplatform.2.2.2.dtd (only an extension version)
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE deploymentplatform…>		

Domain Profile Types, Applicable SCA DTDs and XML header line examples		
Doman Profile File Name	**File Name Extension**	**DTD File**
?xml and DOCTYPE line samples		
Software Package Descriptor (SPD)	spd.xml	softpkg.2.2.2.dtd
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE softpkg …>		
Software Component Descriptor (SCD)	scd.xml	softwarecomponent.2.2.2.dtd (extension version)
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE softwarecomponent …>		
Properties File (PRF)	prf.xml	properties.2.2.2.dtd
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE properties …>		
Device Package Descriptor (DPD)	dpd.xml	devicepkg.2.2.2.dtd
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE devicepkg…>		
Device Configuration Descriptor (DCD)	dcd.xml	deviceconfiguration.2.2.2.dtd
<?xml version="1.0" encoding="UTF-8"?>		

Domain Profile Types, Applicable SCA DTDs and XML header line examples		
Doman Profile File Name	**File Name Extension**	**DTD File**
?xml and DOCTYPE line samples		
<!DOCTYPE deviceconfiguration …>		
DomainManager Configuration Descriptor (DMD)	dmd.xml	domainmanagerconfiguration.2.2.2.dtd (extension version)
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE domainmanagerconfiguration …>		

A. Identify the Domain Profile files and their type used in the OE under test and verify that there is a corresponding DTD file using the above table (OE0588).
   1. **Fail**: Each Domain Profile file type does not have a corresponding DTD.
   2. **Pass**: Each Domain Profile file type has a corresponding DTD.

For each Domain Profile file listed in the table above and found in the above step A, perform steps B, C and D:

B. Verify that the *Domain Profile* files are compliant to the applicable SCA DTD and SCA Extension DTD. (OE0588)
   1. **Fail**: The *Domain Profile* files are not compliant to the DTD files specified in the SCA and SCA Extension DTD.
   2. **Pass**: The *Domain Profile* files are compliant to the DTD files specified in the SCA and SCA Extension DTD.

C. Verify that the *Domain Profile* file has the first two lines with *XML* declaration (?xml) and a *document type declaration* (!DOCTYPE) (OE0590).
   1. **Fail**: The *Domain Profile* file does not have the *XML or DTD* type declarations.
   2. **Pass**: The *Domain Profile* file has the *XML and DTD* type declarations.

D. Verify that each Domain Profile file has the correct file name extension as indicated in the table above (OE0591, OE0592, OE0594, OE0595, OE0596, OE0597, OE0788).
   1. **Fail**: Not all Domain Profile files have the correct file name extension.
   2. **Pass**: Each Domain Profile file has the correct file name extension.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.

2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the Domain Profile files and their type used in the OE under test and verify that there is a corresponding DTD file using the above table (OE0588).				
1. Obtain and record the directory where the Domain Profile (XML) files are located.	**Pass**: One or more directories are located. (OE0588) **Untested:** A directory is not located. (OE0588)		Multiple directories may exist.	

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Identify and record the type of each Domain Profile (XML) file used in the OE. The types are DMD, DCD, DPD, SPD, SCD, PRF and the PDD.	Required types: xxx.dmd.xml (DMD) xxx.spd.xml (SPD) Optional types: xxx.dcd.xml (DCD) xxx.dpd.xml (DPD) xxx.scd.xml (SCD) xxx.prf.xml (PRF) xxx.pdd.xml (PDD) **Pass:** DMD and SPD files found. (OE0588) **Fail:** No DMD or SPD files found. (OE0588)		This can be done by examining the XML file extensions as indicated in the expected results column.	

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Locate and verify the corresponding DTD file from the OE directory for each Domain Profile file type identified in Step 2.	**Pass:** Each type of Domain Profile file has a corresponding DTD file. (OE0588) **Fail:** Not each type of Domain Profile file has a corresponding DTD file. (OE0588)		See table in Test Description for File Name Extension and Matching DTD files.	
For each Domain Profile file listed in the table and found in the above step A, perform steps B, C and D:				
B. Verify that the *Domain Profile* files are compliant to the applicable SCA DTD (OE0588).				
4. Compare *Domain Profile* files from the OE to the format of the SCA v2.2.2 DTD files.	**Pass**: The *Domain Profile* file(s) match(s) the format of the DTD files specified in the SCA. (OE0588) **Fail:** The *Domain Profile* file(s) do not match(s) the DTD files specified in the SCA. (OE0588)		A file comparison tool such as the diff or Cygwin functions in UNIX/Linux or csdiff with windows can be used for this test.	
C. Verify that the *Domain Profile* file has the first two lines with *XML* declaration (?xml) and a *document type declaration* (!DOCTYPE) (OE0590).				

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
5. Confirm that each Domain Profile (XML) file has the *?XML* declaration in the 1st line and *!DOCTYPE* declaration in the 2nd line.	**Pass:** The *Domain Profile* file has the *?XML* and *!DOCTYPE* declarations. (OE0590) **Fail:** The *Domain Profile* file does not have the *?XML* or *!DOCTYPE* declarations. **(**OE0590)		Sample declarations appear in the table in the Test Description <?xml version="1.0" …> <!DOCTYPE …>	
D. Verify that each Domain Profile file has the correct file name extension as indicated in the table above (OE0591, OE0592, OE0 594, OE0595, OE0596, OE0597, OE0788).				
6. Confirm that each *Software Package Descriptor* (SPD) file has a "*.spd.xml*" extension.	**Pass:** Each *Software Package Descriptor* (SPD) file has a "*.spd.xml*" extension. (OE0591) **Fail:** Not each *Software Package Descriptor* (SPD) file has a "*.spd.xml*" extension. (OE0591)			

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
7. Confirm that each *Software Component Descriptor* (SCD) file has a "*.scd.xml*" extension.	**Pass:** Each *Software Component Descriptor* (SCD) file has a "*.scd.xml*" extension. **(**OE0592) **Fail:** Not each *Software Component Descriptor* (SCD) file has a "*.scd.xml*" extension. **(**OE0592)			
8. Confirm that each *Properties Descriptor* (PRF) file has a "*.prf.xml*" extension.	**Pass:** Each *Properties Descriptor* (PRF) file has a "*.prf.xml*" extension. **(**OE0594) **Fail:** Not each *Properties Descriptor* (PRF) files have a "*.prf.xml*" extension. **(**OE0594)			

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
9. Confirm that each *Device Package Descriptor* (DPD) file has a ".dpd.*xml*" extension.	**Pass:** Each *Device Package Descriptor* (DPD) file has a ".dpd.*xml*" extension. **(**OE0595) **Fail:** Not each *Device Package Descriptor* (DPD) file has a ".dpd.*xml*" extension. **(**OE0595)			
10. Confirm that each *Device Configuration Descriptor* (DCD) file has a ".dcd.*xml*" extension.	**Pass:** Each *Device Configuration Descriptor* (DCD) file has a ".dcd.*xml*" extension. **(**OE0596) **Fail:** Not each *Device Configuration Descriptor* (DCD) file has a ".dcd.*xml*" extension. **(**OE0596)			

OE_TC_0185				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
11. Confirm that each *DomainManager Configuration Descriptor* (DMD) file has a ".dmd.*xml*" extension.	**Pass:** Each *DomainManager Configuration Descriptor* (DMD) file has a ".dmd.*xml*" extension. (OE0597) **Fail:** Not each *DomainManager Configuration Descriptor* (DMD) file has a ".dmd.*xml*" extension. (OE0597)			
12. Confirm that each *Deployment Platform Descriptor* (PDD) file has a "*.pdd.xml*" extension.	**Pass:** Each *Deployment Platform Descriptor* (PDD) file has a "*.pdd.xml*" extension. **(**OE0788) **Fail:** Not each *Deployment Platform Descriptor* (PDD) file has a "*.pdd.xml*" extension. **(**OE0788)			
End of Test				

Test Recording Log - OE_TC_185				
Step 1 (directory containing the Domain Profile files)				
Step 2 (Domain Profile file type)	**Step 3** (DTD file)	**Step 4** (Compliant to SCA DTD)	**Step 5** (?XML and !DOCTYPE declarations)	**Steps 6 thru 12** (correct file extensions)
SPD Y/N	SPD Y/N	SPD Y/N	SPD Y/N	"*.spd.xml*" extension Y/N
SCD Y/N	SCD Y/N	SCD Y/N	SCD Y/N	"*.scd.xml*" extension Y/N
PRF Y/N	PRF Y/N	PRF Y/N	PRF Y/N	"*.prf.xml*" extension Y/N
DPD Y/N	DPD Y/N	DPD Y/N	DPD Y/N	"*.dpd.xml*" extension Y/N
DCD Y/N	DCD Y/N	DCD Y/N	DCD Y/N	"*.dcd.xml*" extension Y/N
DMD Y/N	DMD Y/N	DMD Y/N	DMD Y/N	"*.dmd.xml*" extension Y/N
PDD Y/N	PDD Y/N	PDD Y/N	PDD Y/N	"*.pdd.xml*" extension Y/N

Test Summary OE_TC_185

Once testing is complete for every component of the OE under test, report the test result as follows:

Pass: No failures detected

Fail: Failure(s) detected in Step(s) (x). Failure of any associated criteria results in a failure of a requirement.

Untested: Condition which is not testable

N/A: Not Applicable

Overall Test Result (Pass, Fail, Untested, or N/A):

OE0588_____ OE0594_____

OE0590_____ OE0595_____

OE0591_____ OE0596_____

C120 _____ OE0597_____

OE0592_____ OE0788_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____