Joint Tactical Networking Center
Test and Evaluation Laboratory

Software Communications Architecture Version 2.2.2
Manual Operating Environment
Software Test Description

Version 3.3A

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

Change History

Version	Date Modified	Author	Reason for Change
3.3	27 September 2019	JTEL	Initial release
3.3A	13 April 2022	JTEL	Public Release

Table of Contents

LIST OF TABLES

1 Introduction

1.1 Intended Readership

This document, addressed to organizations of Joint Tactical Radio (JTR) Set development and/or T&E responsibility, assumes that the readers are familiar with the objectives and requirements of the Joint Tactical Networking Center (JTNC).

1.2 Purpose

The purpose of this Software Test Description (STD) is to describe the formal manual procedures necessary to verify the software components of the Software Communications Architecture (SCA) v2.2.2 Operating Environment (OE) requirements. A description of the procedures, expected results, and additional comments regarding the test itself is included. There is also space provided to record actual results.

1.3 JTNC and JTEL Overview

The mission of the Joint Tactical Networking Center (JTNC) organization is to develop, integrate and field a family of interoperable, digital, and modular software-defined radios that operate as nodes in a network to ensure secure wireless communications and networking services for mobile and fixed forces. The software defined radio s (Software Defined Radios), based on waveforms and hardware solutions, share a common architecture as defined by the SCA requirements.

JTNC Test and Evaluation Laboratory (JTEL) is the test authority for Compliance Testing against the SCA. The goal of JTEL is to develop Test capabilities and to provide excellence in testing with the highest integrity and quality to SDRs.

The primary mandates of JTEL are the following:

- Support waveform application and operating environment testing for the JTNC based on Interface Control Working Group (ICWG) Approved JTNC Standards.
- Provide compliance test reports and recommendations on the Standards compliance of the tested products to the JTNC for consideration.

Adherence to the SCA allows standardization of the JTR Set infrastructure and is the basis for waveform portability. Portability of waveforms is one of the primary means of reducing development costs and providing for interoperability.

Interoperability is achievable through the reuse of common core waveforms used in all JTR Sets, and through an expanded Defense Information Systems Agency (DISA) Joint Interoperability Test Command (JITC) waveform conformance testing process as a prerequisite to interoperability testing and certification.

1.4 Identification

1.4.1 Scope

Compliance testing of Operating Environment requirements is achievable by executing a combination of the following test methods:

1. Manual – Visual inspection and analysis of artifacts (documents, source code files, eXtensible Markup Language (XML) files and Interface Definition Language (IDL) files).
2. Semi-Automated – Assessment with a limited use of automated testing. For example, using JTAP to cover some parts of the requirements and needing parts of manual test cases to complete the coverage.
3. Automated – Using JTNC Test Application (JTAP), testers have the ability to completely automate verification of some requirements and avoid some manual tests in this document. This test method is outside the scope of this MOESTD document. Refer to the JTAP document for information on this automated test suite. [Reference 8].

1.4.2 Document Overview

This Operating Environment Software Test Description is composed of the following sections:

Section 1: Introduction
This section contains the introduction, the intended audience, an overview of JTNC and JTEL, and the scope of the document.

Section 2: Reference Documents
This section contains the referenced and related documents described by a reference number and its title (includes version and date, if available).

Section 3: Test Preparations
This section describes the procedures necessary for the hardware and software used for the compliance testing of Operating Environment requirements.

Section 4: Test Descriptions
This section describes the test objectives including preconditions, test inputs, expected results, and naming conventions of each test case.

Appendices
Appendix A contains the abbreviations found in this document.

Appendix B contains the manual test steps found in each test case. Because of the large number of test cases in Appendix B, it has been divided into 5 volumes. The volumes are organized by general areas of the SCA. Each volume contains test cases applicable to these areas, as follows:

- Volume 1 - Base Application Interfaces and the OMG Lightweight Log. The Base Application Interfaces include Port, LifeCycle, TestableObject, PropertySet, and Resource.
- Volume 2 - Framework Control Interfaces which include the DomainManager, DeviceManager, ApplicationFactory and Application.
- Volume 3 - Base Device Interfaces which include the Device, LoadableDevice, ExecutableDevice and AggregateDevice.
- Volume 4 - Framework Service Interfaces which include the FileManager, FileSystem and File.

- Volume 5 - AEP and Miscellaneous Requirements.

Appendix C contains the manual test steps for the Application Environment Profile (AEP) Amended requirements. 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 requirement(s) being tested.

Appendix D contains the manual test steps for the SCA Extensions [Reference 10] requirements. 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 requirement(s) being tested.

2 Reference Documents

Referenced documents found within this document include standards, specifications and handbooks.

Table 1: Referenced Documents

Index	Title
1	Software Communications Architecture Specification, v2.2.2, Final, 15 May 2006
2	Software Communications Architecture Specification , Appendix B (Application Environment Profile_Amended (AEP)) Version 2.2.2A, 22 October 2008
3	Software Communications Architecture Specification, Appendix C (Core Framework IDL) Version 2.2.2, Final, 15 May 2006
4	Software Communications Architecture Specification, Appendix D (Domain Profile) Version 2.2.2, Final, 15 May 2006
5	Minimum CORBA Specification, Version 1.0, 02 August 2002
6	Interoperable Naming Service Specifications, Version 1.0.1, November 2000
7	OMG Lightweight Log Service Specification, Version 1.1, February 2005, (formal/05-02-02)
8	JTEL Automated Operating Environment Software Test Description (AOESTD) for SCA v2.2.2. for JTNC Test Application (JTAP), v3.14F, 27 September 2019
9	Software Communications Architecture Specification, Appendix B (Application Environment Profile(AEP)) Version 2.2.2, Final, 15 May 2006
10	Software Communications Architecture Extensions Version 2.2.2, Final, 15 December 2006

3 Test Preparations

Each test case contains the objectives, requirements, manual test steps that direct the tester to execute specific procedures.

3.1 Software Preparation

Preconditions, items that need to be completed or be available before the test has begun, provide the tester a starting point. For example, specific XML files must be available prior to the test. To organize your material, collect the Test Case document, Test Recording Log, and reference material checking that none of the material is missing. Then put all related material together. Knowing about the test will help the organization of the materials and the execution of the test steps.

Supporting documents (see Referenced Documents, section 2) should be available as hard or soft copies.

3.2 Hardware Preparation

The hardware needed for the test is a printer to make hard copies of the supporting documents, mentioned in the previous section. Due to potential sensitive data, laptops may not be available in the lab. The hardcopies or documents contained on CDs are useful in these situations.

Additional test resources may include laptops or other computers with compatible applications (i.e., MS Word, Adobe Acrobat) to view file content, compilers to check coding standards, and context sensitive search software or XML parsers to search keywords or other text. PrismTechTM's Spectra tool is required for testing the Domain Profile.

4 Test Descriptions

4.1 Test Objective

A test objective describes what the object of performing the test case is. It should be as specific as possible and consist of a list of verifiable objectives for the test case.

4.2 Preconditions

A precondition is a condition or predicate that must always be true just prior to the execution of the test. A violation of the precondition may affect the outcome of the test.

The Manual Operating Environment test cases have these preconditions in common:

- All the Domain Profile files of the OE are available
- The Domain Profile tests for the OE have passed
- The source code for the component under test is available

4.3 Test Inputs

Each test case requires specific test inputs that verify the requirements. The first source of the test input comes from the IDL Reference section of each test case. The name and description (behavior and synopsis) input and output variables of each component come directly out of the SCA v2.2.2 Specification [Reference 1]. A second source of the test inputs for most test cases are the source code files provided. Use the Places to Verify section to assist in identifying the specific source code for a given test case. Another source of the test input comes from the column Comments, seen in the Manual Test Steps. Certain conditions will direct the tester to locate specific files (source code, XML, executable, etc.) required for verification of a requirement. A third source of test inputs are the keywords or text strings used in searches to locate operations and arguments found in source code files.

4.4 Expected Results

Expected Results of the test cases defines the conditions for passing the step. A Test Recording Log provides the tester an area to write down intermediate test results while executing the test steps. The tester will enter actual results into the Test Recording Log.

4.5 Definitions of Test Results

The definitions for the various test results are as follows:

1. **N/A**: The requirement is not applicable to this component.
2. **Untested**: The test could not be completed because some necessary information is missing. If the test is a final test and the developer cannot provide the missing information then this becomes a failure.
3. **Fail**: The expected results of a step were not met. Normally this means a requirement fails.
4. **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.

4.6 OE Test Cases

4.6.1 Naming Conventions

Each test case below uses this convention - test case number followed by the functional area and its test objective. The test case number consists of the three components mentioned in Table 2.

Table 2: Test Case Numbering Conventions

Abbreviation	Meaning
OE	Operating Environment
TC	Test Case
xxx	Unique 3-digit numerical identifier

The Test Objective, an expression of facts or conditions as perceived without interpretations or judgments, follows this format:

- First sentence starts with "*This test case verifies* <***requirement number(s)***>"
- Second sentence starts with "*The objective of this test is to* verify <***explanation***>".

4.6.2 SCA v2.2.2 OE Manual Test Case Objectives (Appendix B)

4.6.2.1 OE_TC_001 - CORBA :: NamingService

This test case verifies OE0007 and OE0746. The objective of this test is to verify that the OE provides an implementation of a CORBA *Naming service* and implements the *CosNaming* module *NamingContext* interface operations: *bind*, *bind_new_context*, *unbind*, *destroy* and *resolve*.

4.6.2.2 OE_TC_002 - CORBA :: Log Producer

This test case verifies OE0011, OE0012, OE0013 and OE0747. The objective of this test is to verify that the OE outputs proper log records.

4.6.2.3 OE_TC_003 - Event Service

This test case verifies OE0066, OE0067, and OE0702. The objective of this test is to verify that two event channels, named IDM_Channel and ODM_Channel, are created.

4.6.2.4 OE_TC_004 - Device :: allocateCapacity

This test case verifies OE0407. The test will verify that the *allocateCapacity* operation sets the *usageState* attribute to ACTIVE, when capacity is being used and any capacity is still available for allocation.

4.6.2.5 OE_TC_005 - Log Service :: LogStatus

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C182, OE0700-C183, OE0700-C184, and OE0700-C185, of the OMG Lightweight Log [Reference 7]. The objective of this test is to verify that the *get_max_size()*, *set_max_size()* and *get_current_size()*

operations perform correctly according to the Lightweight Log Service Specification, which is cited in the SCA v2.2.2. The *set_max_size()* operation must be able to handle valid and invalid input from its parameter in order to set the maximum size of the Log. The *get_max_size()* operation must return the maximum size of the Log. The *get_current_size()* must return the size, in bytes, of the records that currently occupy the logging storage area.

4.6.2.6 OE_TC_006 - Log Service :: get_n_records

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C186, of the OMG Lightweight Log [Reference 7]. The objective of this test is to verify that the *get_n_records()* operation returns the number of records currently stored in the log storage area.

4.6.2.7 OE_TC_007 - Log Service :: LogFullAction

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C187, OE0700-C188, OE0700-C189, OE0700-C190, and OE0700-C191. The objective of this test is to verify that the *get_log_full_action*() operation returns the action set by the *set_log_full_action*() operation and that the *set_log_full_action*() operation properly validates and sets the LogFullAction and availability status logFull.

4.6.2.8 OE_TC_008 - Log Service :: get_availability_status

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C192. The objective of this test is to verify that the existence of get_availability_status is confirmed and its return value is verified.

4.6.2.9 OE_TC_009 - Log Service :: administrative_state

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C193, OE0700-C194, OE0700-C195, OE0700-C196, OE0700-C197, and OE0700-C198 of the OMG Lightweight Log [Reference 7]. The objective of this test is to verify that the *get_administrative_state*() operation returns the action set by the *set_administrative_state()* operation.

4.6.2.10 OE_TC_010 - Log Service :: get_operational_state

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C199 and OE0700-C200 of the OMG Lightweight Log [Reference 7]. The objective of this test is to verify that the *get_operational_state()* operation returns the actual state of the log; the possible values are ENABLED or DISABLED.

4.6.2.11 OE_TC_011 - Log Service :: get_record_id_from_time

This test case partially verifies the log service requirement, OE0700 by testing the criteria OE0700-C201 and OE0700-C202, The objective of this test is to verify that log records can be retrieved by time.

4.6.2.12 OE_TC_012 - Log Service :: retrieve_records

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C203, OE0700-C204, OE0700-C205, OE0700-C206, and OE0700-C207. The objective of this test is to verify that the *retrieve_records* operation meets the specifications of the OMG Lightweight Log [Reference 7]. By examining source code files in the application, the test engineer shall verify that the

currentId returns the ID of the starting records, the *howMany* contains the number of records returned, and *LogRecordSequence* contains the sequence of the retrieved records.

4.6.2.13 OE_TC_013 - Log Service :: retrieve_records_by_level

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C208, OE0700-C209, OE0700-C210, OE0700-C211, OE0700-C212, and OE0700-C213. The objective of this test is to verify that the *retrieve_records_by_level* operation meets the specifications of the OMG Lightweight Log [Reference 7]. By examining the executable source code files in the application, the test engineer shall verify that the *currentId* returns the ID of the starting records, the *howMany* contains the number of records returned, and *LogRecordSequence* contains the sequence of the retrieved records.

4.6.2.14 OE_TC_014 - Log Service :: retrieve_records_by_producer_id

This test case verifies OE0700, OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, and OE0700-C219. The objective of this test is to verify that the operating *retrieve_records_by_producer_id* of the log service complies with the OMG Lightweight Log [Reference 7].

4.6.2.15 OE_TC_015 - Log Service :: retrieve_records_by_producer_name

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C220, OE0700-C221, OE0700-C222, OE0700-C223, OE0700-C224, and OE0700-C225. The objective of this test is to verify that log records can be retrieved using a set of producer names.

4.6.2.16 OE_TC_016 - Log Service :: write_records

This test case verifies the log service requirement, OE0700, partially by testing the criteria: OE0700-C226, OE0700-C227, OE0700-C228, OE0700-C229, OE0700-C230, and OE0700-C240. The objective of this test is to verify that the *write_records ()* operation meets the specifications of the OMG Lightweight Log [Reference 7]. This test case will verify that the *write_records()* operation adds the log records supplied in its parameter to the Log, inserts the UTC time into the time field, and assigns a unique record id to the id field of the *LogRecord*. If there is insufficient storage in the Log when *write_records()* is performed, it must check the *LogFullAction* status and modify its behavior accordingly. These Requirements must be fulfilled to uphold the SCA v2.2.2 concerning the *write_records()* operation.

4.6.2.17 OE_TC_017 - Log Service :: write_record

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235. The objective of this test is to verify that the write record of the log service complies with the OMG Lightweight Log [Reference 7]. The *write_record* operation adds a single record to the log for later retrieval. The *write_record* operation automatically adds a timestamp and unique identifier and must accommodate the conditions where the log becomes full (allowing for a user selected WRAP or HALT action).

4.6.2.18 OE_TC_018 - Log Service :: clear_log

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C236 and OE0700-C237 of the OMG Lightweight Log [Reference 7]. It also tests OE0700-C186,

get_n_records(). The objective of this test is to verify that the *clear_log0* operation removes all the logging records from the storage area, does not change the size of the logging area, and sets the availability status *logFull* state to false.

4.6.2.19 OE_TC_019 - Log Service :: destroy

This test case partially verifies the log service requirement, OE0700 by testing the criteria, OE0700-C238, and OE0700-C239. The objective of this test is to verify that existing records in the log storage area are not recoverable once deleted and that memory resources associated with the log storage area are released.

4.6.2.20 OE_TC_020 - Port :: connectPort

This test case verifies requirement, OE0069, and the supporting criterion, OE0069-C002 and requirement OE0070 and supporting criterion OE0070-C004. The objective of this test is to verify that the *connectPort* operation makes the connection to the component identified by its input parameters, the connection parameter and the connectionId parameter. The connectionId is associated to the two ports connected by the connectPort operation as the unique identifier to be used by the disconnectPort operation when breaking this specific connection. Furthermore, this test verifies that the *connectPort* operation raises the InvalidPort exception when the input connect parameter is an invalid object reference for the port. The requirement OE0070 upholds the criterion, OE0070-C004, that the *connectPort* operation populates the InvalidPort exception with an errorCode of 1 when the input cannot be narrowed to the appropriate object reference.

4.6.2.21 OE_TC_021 - AEP Applications have no abnormal termination

This test case verifies OE0798, which is equivalent to OE0661 in the base SCA v2.2.2 Appendix B [Reference 9], SCA AEP. For product lines that implement the AEP Amended, OE0798 is evaluated in lieu of OE0661 for the same objective. Table B-6 of the AEP Amended specifies the mandatory POSIX Signals function to be provided by the OE. Additionally, the SCA AEP Amended [Reference 2] specifies that the signals generated by these signal functions when called must be handled properly so that other processes are not impacted because certain RTOSs do not support multiple processes. The objective of this test case is to verify that the OE, which is the provider of these signals functions, implements signals handlers for each signal generated by the signal functions in order to properly manages process termination.

4.6.2.22 OE_TC_022 - AggregateDevice devices

This test case verifies OE0459. The objective of this test is to verify that the readonly devices attribute will contain a list of devices that have been added to this device or a sequence length of zero if the device has no aggregation relationships with other devices.

4.6.2.23 OE_TC_023 - AggregateDevice :: addDevice

This test case verifies OE0460. The objective of this test is to verify that the *addDevice* operation will add the input *associatedDevice* parameter to the *AggregateDevice's* devices attribute when the *associatedDevice* does not exist in the devices attribute. The purpose of the addDevice is to provide the mechanism to associate a device with another device.

4.6.2.24 OE_TC_024 - Domain Manager :: registerDeviceManager

This test case verifies OE0732. The objective of this test is to verify that an InvalidProfile exception is raised by the DomainManager when a DeviceManager's DCD file or any file referenced within the DCD file does not exist.

4.6.2.25 OE_TC_025 - AggregateDevice :: addDevice FAILURE_ALARM

This test case verifies OE0461. The objective of this test case is to verify that the *addDevice* operation writes a FAILURE_ALARM log record when an associatedDevice cannot be successfully added to the *AggregateDevice's devices* attribute.

4.6.2.26 OE_TC_026 - Domain Manager :: installApplication raises ApplicationAlreadyInstalled

This test case verifies OE0722. The objective of this test is to verify that application installations are properly tracked by the DomainManager, that an attempt to install an already installed application is properly detected and that the appropriate exception is raised.

4.6.2.27 OE_TC_027 - AggregateDevice :: addDevice raises InvalidObjectReference

This test case verifies OE0462. The objective of this test case is to verify that the *addDevice* operation raises the CF *InvalidObjectReference* when the input *associatedDevice* parameter is a nil CORBA object reference.

4.6.2.28 OE_TC_028 - AggregateDevice :: removeDevice

This test case verifies OE0463. The objective of this test is to verify that the *removeDevice* operation provides the mechanism to disassociate a device from another device. Using the input parameter *associatedDevice*, the *removeDevice* operation will remove devices from the *AggregateDevice device* attribute.

4.6.2.29 OE_TC_029 - AggregateDevice :: removeDevice FAILURE_ALARM

This test case verifies requirement OE0464. The AggregateDevice functionality permits child device(s) to be associated with a parent device (aggregate) and the *removeDevice()* operation allows for the disassembly of the aggregate. The objective of this test is to verify that an AggregateDevice can be properly disassociated, that failure during a *removeDevice()* operation results in the writing of a FAILURE_ALARM log record.

4.6.2.30 OE_TC_030 - AggregateDevice :: removeDevice raises InvalidObjectReference

This test case verifies requirement OE0465. The objective of this test is to verify that the *removeDevice* operation raises the CF InvalidObjectReference when (1) the input associatedDevice parameter is a nil CORBA object reference or (2) the associatedDevice parameter does not exist in the AggregateDevice devices attribute.

4.6.2.31 OE_TC_031 - FileManager :: list

This test case verifies OE0528, OE0542, OE0543, OE0582, OE0583, OE0736, and OE0755 for a *FileManager*. The objective of this test is to verify that the *FileManager* correctly implements or delegates the *list* operation to the mounted *FileSystems* (OE0582) when supporting the "*" and "?" wildcard characters (OE0542) in the search parameter, checking that certain other requirements are

also fulfilled (OE0528, OE0543, OE0583, OE0736, OE0755). The *FileManager*'s *list* operation must return a FileInformationSequence matching the search pattern specified in the input parameter (OE0543), returning the name, kind and size for each file found (OE0528). The test case verifies that the wildcard patterns, "*" and "?", will only be applied following (to the right of) the farthest right-most forward-slash character in the input pattern parameter (OE0736). The test case also verifies that the *FileManager*'s *list* operation will return a zero length sequence if the search pattern is not found (OE0755). In addition, the test case must verify that the mounted *FileSystem* name is removed from the pathname of the input pattern parameter in the conditional case that the *FileManager* delegates the list operation to the mounted *FileSystem*(s) (OE0583).

4.6.2.32 OE_TC_032 - Port :: connectPort raises OccupiedPort

This test case verifies requirement, OE0071. The objective of this test is to verify that the *connectPort* operation raises the OccupiedPort exception if a connection is attempted when the port is fully occupied.

4.6.2.33 OE_TC_033 - DeviceManager Register

This test case verifies OE0473. The objective of this test is to verify that all device managers register with a domain manager when the device manager starts.

4.6.2.34 OE_TC_034 - DeviceManager Register and the DCD file

This test case verifies OE0466, OE0467, and OE0474. The objective of this test is to verify that the Device Manager's identifier is unique and is retrieved from the Device Manager's DCD. It will also verify that the other information from the DCD is used correctly to determine Services, Devices, mount point, and attributes of a Device Manager. Note that an id attribute in the DCD is required to be a DCE UUID this test will verify that also.

4.6.2.35 OE_TC_035 - Port :: disconnectPort

This test case verifies OE0072, OE0073 and OE0073-C003. The test will verify that a connection to the component identified by the input *connectionId* parameter is broken when the *disconnectPort* operation is executed. The test will also verify that when the Port *disconnectPort* operation through the Core Framework is executed that it throws an InvalidPort exception when the connectionId parameter is not a known connection. Finally, the test will verify that when the exception is raised, it contains an error value of 2 and a message string. An error code of 2 means the Port name was not found. Though the criterion indicates that there are 2 valid values for the exception, only error code 2 is valid for the *disconnectPort* operation.

4.6.2.36 OE_TC_037 - LifeCycle :: initialize raises InitializeError

This test case verifies OE0074. The objective of this test is to verify that the *InitializeError* exception is raised by the initialization operation when an initialization error occurs.

4.6.2.37 OE_TC_038 - LifeCycle :: releaseObject

This test case verifies OE0075. The objective of this test is to verify that all memory allocated during instantiation and the lifetime of an object is released before or while the object is being destroyed.

4.6.2.38 OE_TC_039 - LifeCycle :: releaseObject raises ReleaseError

This test case verifies OE0078. The objective of this test is to verify that the CF::LifeCycle::R*eleaseError* exception is raised by the CF::LifeCycle::r*eleaseObject* operation when a release error occurs.

4.6.2.39 OE_TC_040 – Exceptions from the Mounted File System

This test case verifies requirement OE0739. The objective of this test will determine if the File Manager interface propagates or re-raises the exceptions just as if the File System were accessed directly. Since FileManager is a class that inherits from FileSystem, it is possible that some of the operations are implicitly inherited from FileSystem; for such operations the exceptions raised are per the FileSystem exceptions (and requirement OE0739 is considered fulfilled). Where the FileManager redefines operations, the exceptions should be propagated (either caught and then re-raised, or left uncaught) in order to fulfill requirement OE0739. There are three exceptions inherited from the File System that the File Manager must raise (InvalidFileName, UnknownFileSystemProperties, and FileException).

4.6.2.40 OE_TC_041 - TestableObject :: runTest uses testId parameter

This test case verifies requirement OE0079. The objective of this test is to verify that the *runTest* operation input *testId* parameter determines which of its predefined test implementations are performed.

4.6.2.41 OE_TC_042 - TestableObject :: runTest uses testValues parameter

This test case verifies requirement OE0080. The objective of this test is to verify that the id/value pair(s) of the *testValues* parameter is/are used to supply additional information whenever needed for the implementation-specific test to be run.

4.6.2.42 OE_TC_043 - TestableObject :: runTest returns results

This test case verifies requirement OE0081. The objective of this test is to verify that the *runTest* operation returns the result(s) of each test in the testValues parameter for the components of the Core Framework that have defined testable properties.

4.6.2.43 OE_TC_044 - AEP file mode creation masks

This test case verifies OE0666. The objective of this test is to verify that the file mode creation mask for any object created is at a minimum S-IRWXU. This mask says that the creator, i.e., owner, will have read, write, and execute rights to the object created. The file mode creation mask is used for the creation of directories and files.

4.6.2.44 OE_TC_045 - TestableObject :: runTest validates testValues parameter

This test case verifies requirement OE0083. The objective of this test is to verify that all testValues parameter properties that are used by the runTest*()* operation are validated.

4.6.2.45 OE_TC_046 - TestableObject :: runTest validates parameters

This test case verifies requirement OE0084. The objective of this test is to verify that the *runTest* operation will not execute any testing for a given component when the testId/testValue pair is not known by the component or is out of range.

4.6.2.46 OE_TC_047 - PropertySet :: configure

This test case verifies OE0091. The objective of this test is to verify that the *configure* operation of the component assigns the values indicated by its input configProperties parameter to the corresponding ids of the component properties.

4.6.2.47 OE_TC_048 - PropertySet :: configure property minimums

This test case verifies OE0092. The objective of this test is to verify that the *configure* operation of a component must, at a minimum, support the readwrite and writeonly properties which are referenced in its SPD. The SPD indirectly references the properties through references to PRF files.

4.6.2.48 OE_TC_049 - PropertySet :: configure raises PartialConfiguration

This test case verifies OE0093. The objective of this test is to verify that the *PartialConfiguration* exception is raised by the configure operation when some configuration properties were successfully set and some configuration properties were not successfully set. The configuration property being configured is the *ID* found in the *simpl*e element of the PRF file.

4.6.2.49 OE_TC_050 - Resource :: start raises StartError

This test case verifies OE0105 and OE0099. The test will verify that when the *Resource start* operation through the Core Framework is executed that it throws a StartError exception when an error occurs while starting a resource. In addition, the test will verify that the exception contains an error number of the type CF ErrorNumberType.

4.6.2.50 OE_TC_051 - Domain Manager :: unregisterService

This test case verifies requirement OE0327. The objective of this test is to verify that the *unregisterService* operation removes the unregisteringService entry specified by the input name parameter from the DomainManager.

4.6.2.51 OE_TC_052 - Domain Manager :: unregisterService client-side

This test case verifies OE0328. The objective is to test that the DomainManager releases the service on the client-side as a result of the *unregisterService* operation.

4.6.2.52 OE_TC_053 - Domain Manager :: unregisterDeviceManager

This test case verifies OE0274. The objective of this test is to verify that the *unregisterDeviceManager* operation releases all device(s) and service(s) associated with the device manager that is being unregistered. Even though devices and services are found in the Device Configuration Descriptor file, it is not necessary to identify them. Reviewing the source code is sufficient to verify that the devices and services are release from the associated device manager.

4.6.2.53 OE_TC_054 - Domain Manager logs defined in DMD file

This test case verifies OE0217. The objective of this test is to verify that all logs used by the DomainManager are defined in the DMD.

4.6.2.54 OE_TC_055 - Domain Manager :: unregisterService raises UnregisterError

This test case verifies OE0337 and OE0202. The objective of this test is to verify that DomainManager's *unregisterService* operation will raise an *UnregisterError* exception when an unsuccessful unregistration occurs. This *UnregisterError* exception provides information about the error, such as the error number that is a CF ErrorNumberType value.

4.6.2.55 OE_TC_056 - Domain Manager services defined in DMD file

This test case verifies requirement, OE0218. The objective of this test case is to verify that the domain manager will only allow a service listed in the DomainManager Configuration Descriptor(DMD) file to be made available for use after it has been successfully registered by the *registerDeviceManager* or *registerService* operations.

4.6.2.56 OE_TC_057 - Domain Manager :: installApplication raises ApplicationInstallationError

This test case verifies OE0200, OE0261 and OE0268. The objective of this test is to verify the error handling of the domain manager's *installApplication* operation when installation of the application's files does not succeed. Furthermore, verify the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.57 OE_TC_058 - Device operationalState

This test case verifies OE0400 and the supporting criteria: OE0400-C092, OE0400-C093, OE0400-C094, OE0400-C095, and OE0400-C096. The objective of this test case is to verify that upon the occurrence of a change of operationalState of the device, a StateChangeEventType event is sent to the Incoming Domain Management event channel. The StateChangeEventType event must contain current values for *producerId, sourceId, stateChangeCategory, stateChangeFrom* and *stateChangeTo* field.

4.6.2.58 OE_TC_059 - Mandatory interfaces of the AEP

This test case verifies OE0001. The objective of this test is to verify that the OE provides, at a minimum, the options and functions designated as mandatory in Appendix B of the SCA [Reference 9].

4.6.2.59 OE_TC_060 - FileSystem Filename Lengths

This test case verifies OE0002, OE0605, OE0733, OE0605-C123, OE0605-C124, and OE0733-C125. The objective of this test is to verify that the file systems of the OE will support a filename length of 40 characters, a path length of 1024 characters, and use valid characters. In this test case, filename means both directory name and filename.

4.6.2.60 OE_TC_061 - Minimum CORBA

This test case verifies OE0003. The objective of this test is to verify that the functionality of minimum CORBA, as described in Minimum CORBA Specification Version 1.0: OMG Document [Reference 5] formal/02-08-01, August 2002, is met by the OE middleware.

4.6.2.61 OE_TC_062 - Base Device Interfaces

This test case verifies OE0701. The objective of this test is to verify that Base Device Interfaces are implemented using the CF IDL as presented in Appendix C [Reference 3]. The CF IDL implements the CF (Core Framework) module.

4.6.2.62 OE_TC_063 - CORBA :: CosEventComm

This test case verifies OE0063, OE0064, and OE0065. The objective of this test is to verify that if the component consumes or produces events, it must follow the respective interface(s) from CosEventComm.

4.6.2.63 OE_TC_064 - Resource :: stop raises StopError

This test case verifies OE0103 and OE0100. The objective of this test is to verify that the *stop* operation will raise an exception, with the proper data attached, when an error occurs during the stopping of a resource.

4.6.2.64 OE_TC_065 - Domain Manager :: registerService

This test case verifies OE0314. The objective of this test case is to verify that the *registerService* operation associates its *registeringService* parameter with the device manager with which it is registered. The registered device manager that the *registeringService* must be associated with is indicated by the *registeredDeviceMgr* parameter.

4.6.2.65 OE_TC_066 - File :: read raises IOException

This test case verifies OE0515 and OE0507. The test will verify that when the File *read* operation through the Core Framework is executed that it throws an IOException exception when a read error occurs. In addition, the test will verify that the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.66 OE_TC_067 - File :: close

This test case verifies OE0522 and OE0523. The test will verify that when the *close* operation through the CF::File interface is executed that it releases any OE file resources associated with the component. Furthermore, this test will verify that the *close* operation makes the particular file object unavailable to the component.

4.6.2.67 OE_TC_068 - File :: close raises FileException

This test case verifies OE0524 and OE0598. The test will verify that the File::*close* operation raises the CF::FileException error when it cannot successfully close a file. It also verifies that the exception contains the error number.

4.6.2.68 OE_TC_069 - File :: setFilePointer raises FileException

This test case verifies OE0526 and OE0598. The objective of this test is to verify that when the *setFilePointer* operation is executed, it raises the CF::*FileException* when the file pointer for the referenced file cannot be set to the value of the input *filePointer* parameter. The SCA states, "The *setFilePointer* operation positions the file pointer where the next read or write will occur". The *filePointer* attribute of the *setFilePointer* operation gets set with an input value.

4.6.2.69 OE_TC_070 - Domain Profile

This test case verifies OE0588, OE0590, OE0591, OE0592, OE0594, OE0595, OE0596, and OE0597. The objective of this test is to verify that the Domain Profile is compliant to the Document Type Definitions (DTD) provided in Appendix D [Reference 4]. The tests also verify 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.

4.6.2.70 OE_TC_071 - Resource :: stop

This test case verifies OE0102. The objective of this test is to verify that the *stop* operation properly disables a resource and puts it in a non-operating condition.

4.6.2.71 OE_TC_072 - TestableObject :: runTest parameters

This test case verifies requirement OE0082. The objective of this test is to verify that components implementing the *runTest* operation have at least the *testIds* and *testValues* specified in the Properties Descriptor file (PRF). The PRF details component and device attribute settings. The *test* element is used to specify a list of test properties for executing the *runTest* operation in order to perform a component specific test. This element contains *inputvalue* and *resultvalue* elements and it has an *id* attribute *for* grouping test properties to a specific test. The *id* attribute will be represented by a numeric value, the i*nputvalue* is used to configure the test to be performed, and the *resultvalue* contains the results of the tests.

4.6.2.72 OE_TC_073 - DomainManager :: registerService

This test case verifies OE0725. The objective of this test is to verify that if a new service has the same name and type as a previously registered service, but the object of the registered service does not exist, then the new service will be registered.

4.6.2.73 OE_TC_074 - ApplicationFactory :: create

This test case verifies requirement OE0184. The objective of this test is to verify that if the specified event channel does not already exist, then the *create* operation will create the specified event channel.

4.6.2.74 OE_TC_075 - Provided interfaces described as provides ports

This test case verifies OE0614. The objective of this test is to verify that interfaces provided by a component are described the component's Software Component Description (SCD) file as *provides* ports.

4.6.2.75 OE_TC_076 - Required interfaces described as uses ports

This test case verifies OE0615. The objective of this test is to verify that interfaces required by a component are described in the component's Software Component Description (SCD) file as *uses* ports.

4.6.2.76 OE_TC_077 - SCA APIs and non-IDL interfaces

This test case verifies OE0741, OE0742 and OE0743. The objective of this test is to verify that another engineering group can address any interface and data exchange issues within and between operating environment and applications during a later maintenance software cycle. The test depends

upon the usability of the available Interface Control Documents (ICD) for the non-standard interfaces. Therefore, all APIs implemented in a JTR set must have their interfaces generated with an IDL. Additionally, all IDL interfaces that are not defined by SCA documents (SCA v2.2.2, APIs) need be accurately and thoroughly described using a formal service definition in the ICDs of the Core Framework. This test case also checks that APIs generated using methods other than IDL must have an IDL mapping in the service definition.

4.6.2.77 OE_TC_079 - Device :: usageState

This test case verifies OE0345, OE0381 and its supporting criteria. The objective of this test case is to verify the device's usageState attribute. Furthermore, this test case is to verify that a device sends a StateChangeEventType event to the Incoming Domain Management event channel(IDM_CHANNEL) when the *usageState* attribute is changed by the *allocateCapacity* and *deallocateCapacity* operations. The test case must also verify the criteria associated with this requirement. The criteria ensure that the StateChangeEventType event is populated with the *producerId, sourceId, stateChangeCategory* as USAGE_STATE_EVENT, *stateChangeFrom* field and *stateChangeTo* field.

4.6.2.78 OE_TC_080 - Device :: adminState commanded to be LOCKED

This test case verifies requirements OE0386, OE0387, OE0388 and OE0389. The objective of this test case is to verify

- that the adminState attribute contains a device's admin state (OE0386),
- that the adminState attribute may only be set to LOCKED or UNLOCKED, and
- that setting the adminState attribute to its current value (LOCKED or UNLOCKED) does not cause any processing to occur (OE0387).

This test case also verifies that the *adminState* attribute (of the device), upon being commanded to be LOCKED, transitions from the UNLOCKED to the SHUTTING_DOWN state and sets the adminState to LOCKED for all its aggregated (i.e. children) devices (OE0388). Furthermore, this test case verifies that the adminState of an aggregate (i.e. parent) device transitions to the LOCKED state only after all of its aggregated devices are LOCKED and its usageState is IDLE (OE0389). If the device is not an aggregate device, then this test case must ensure that the usageState is IDLE before the adminState transitions into the LOCKED state (OE0389).

4.6.2.79 OE_TC_081 - Device :: allocateCapacity

This test case verifies requirement, OE0405. The objective of this test case is to verify that the *allocateCapacity* operation reduces the capacities of the device using the allocation capacities specified in the input parameter. The properties for the allocation capacities should be defined in the XML. The *allocateCapacity* operation should always check that the device's adminState is UNLOCKED, the device's operationalState is ENABLED, and the device's usageState is not BUSY before it reduces the current capacities of the device.

4.6.2.80 OE_TC_082 - Device :: allocateCapacity BUSY

This test case verifies OE0406. The objective of this test is to verify that the *allocateCapacity* operation of the device can accurately determine when it is not able to allocate any further

capacity. Capacities for a device are usually specified by properties referenced in the DCD and SPD files. The *allocateCapacity* operation must set the usageState attribute to BUSY when the *allocateCapacity* operation determines that no further capacity can be allocated.

4.6.2.81 OE_TC_083 - Device :: allocateCapacity Failure

This test case verifies OE0408. The objective is to determine whether an allocation request has the possibility of failing* and if it can and if it does, then verify that the requirement to return FALSE is met by the *allocateCapacity* operation.

4.6.2.82 OE_TC_084 - Device :: deallocateCapacity

This test case verifies requirement OE0411. The objective is to test that capacities are properly deallocated within the *deallocateCapacity()* operation.

4.6.2.83 OE_TC_085 - Device :: deallocateCapacity usageState

This test case verifies requirements OE0412, OE0413, and OE0414. The objective is to test that the *deallocateCapacity* operation properly sets the *usageState* and *adminState*. The usageState definition allows for a device to be Active, but still have additional capacity available – it is conceivable that an SCA-compliant device could be constructed for which all of its component capacities are allocated in their entirety, such a device could be BUSY or IDLE, but never ACTIVE. The adminState indicates the permission to use or prohibition against using the device; it can be set to LOCKED and UNLOCKED values, and transiently holds the value of SHUTTING_DOWN.

4.6.2.84 OE_TC_086 - Device :: deallocateCapacity raises InvalidCapacity

This test case verifies OE0415. The objective of this test is to verify that the *deallocateCapacity* operation will raise the InvalidCapacity exception, when the capacity ID is invalid or the capacity value is the wrong type.

4.6.2.85 OE_TC_087 - Device :: releaseObject

This test case verifies OE0420. The objective of this test is to verify that the *releaseObject* operation will cause the Device to be unavailable when the Device's adminState transitions to LOCKED.

4.6.2.86 OE_TC_088 - Device :: adminState attribute changes

This test case verifies OE0390 and its criteria, OE0390-C087, OE0390-C088, OE0390-C089, OE0390-C090, and OE0390-C091. The objective of this test is to verify that a state change event is sent to the Incoming Domain Management channel (IDM_CHANNEL) when a device's state changes and that all the required fields in the state change event message are set correctly.

4.6.2.87 OE_TC_089 - Device :: releaseObject raises ReleaseError exception

This test case verifies OE0423. The objective of this test is to verify that a *releaseObject* will raise the *ReleaseError* exception when an internal processing error prevents successful release of a logical device.

4.6.2.88 OE_TC_090 - DeviceManager :: getComponentImplementationId

This test case verifies requirement OE0504 and OE0505. The objective of this test is to verify that the *getComponentImplementationId* operation returns the SPD implementation element's id attribute that matched the element used to create the component identified by the input parameter. Furthermore, this test verifies that the returned value of the *getComponentImplementationId* operation is an empty string when the input parameter does not match any SPD implementation element used to create the component.

4.6.2.89 OE_TC_091 - Framework Services Interfaces

This test case verifies requirement OE0506. The objective of this test is to see whether the CF IDL and the CF source code are a match for the CF IDL that is in the SCA v2.2.2 Appendix C [Reference 3].

4.6.2.90 OE_TC_092 - Device :: Logical Device executable parameters

This test case verifies OE0618 and OE0619. The objective of this test is to verify that the logical device which are also executable devices accept the standard argv arguments as described by the POSIX exec family of functions (OE0618), and also verifies that the ordering of the argv arguments is in the format that is specified in the SCA v2.2.2, Section 3.1.3.3.3.5.1.3(OE0619).

4.6.2.91 OE_TC_093 - TestableObject :: runTest exception parameter

This test case verifies OE0087. The objective of this test is to verify that when the *runTest* operation is called with invalid testValues, the id(s) of the invalid testValues are returned as the invalidProperties parameter of the UnknownProperties exception. Invalid test values are values that are not known by the component or values that are out of range.

4.6.2.92 OE_TC_094 - Device :: Logical Device - CORBA

This test case verifies OE0620. This test will verify that a logical device is limited to using CORBA as defined in the minimumCORBA specification [Reference 5] and CORBA services as defined in the SCA specification [Reference 1]. The CORBA services defined here are naming service, event service, and the optional log service. A logical device is a software component that implements one of the Basic Device Interfaces, namely, Device, LoadableDevice, ExecutableDevice, and AggregateDevice.

4.6.2.93 OE_TC_095 - Device :: Logical Device - CORBA Register

This test case verifies OE0621. The test will verify that a logical device will register itself with a device manager using the parameter *DEVICE_MGR_IOR*. A logical device is a software component that implements one of the Basic Device Interfaces, namely, Device, LoadableDevice, ExecutableDevice, and AggregateDevice.

4.6.2.94 OE_TC_096 - Aggregate Device

This test case verifies OE0622. The objective of this test is to verify that a child device will add itself to a parent device using the executable Composite Device IOR parameter per SCA 3.1.3.2.4.5. Section 3.1.3.2.4.5 explains how the Composite_Device_IOR is obtained from the entry point of the device in order for the child device to add itself to the aggregate device of the parent.

4.6.2.95 OE_TC_097 - Device :: Logical Devices - CF Interfaces

This test case verifies OE0623. The objective of this test is to verify that the values associated with the parameters (PROFILE_NAME, COMPOSITE_DEVICE_IOR, DEVICE_ID and DEVICE_LABEL) as described in 3.1.3.2.4.5 are used to set the Device's softwareProfile, compositeDevice, identifier, and label attributes, respectively. Since these parameters are execute operation parameters, the devices being tested must be executable devices.

4.6.2.96 OE_TC_098 - Device :: Logical Device allocation properties

This test case verifies OE0627. The objective of this test is to verify that a logical device defines all of the allocation properties that it uses in property descriptor files that are referenced in the device's SPD file. An id of an allocation property should be a Distributed Computing Environment Universal Unique Identifier (DCE UUID). All allocation property ids defined by the logical device in the property descriptor files referenced by the SPD must be supported by the *allocateCapacity* and *deallocateCapacity* operations in the logical device. The *allocateCapacity* and *deallocateCapacity* operations must support additional allocation properties other than those listed in the device's Domain Profile files (SPD, PRF).

4.6.2.97 OE_TC_099 - General Rules : Higher Order Language

This test case verifies OE0628. The objective of this test is to verify through inspection that new OE Software developed for an SCA-compliant system is developed in a standard higher order language.

4.6.2.98 OE_TC_100 - Legacy Software interfaces

This test case verifies OE0630. The objective of this test is to verify that legacy software components of the Operating Environment that interface with the Core Framework are SCA compliant. Even though the term "legacy" is not defined in the SCA document, the generally accepted definition of the term is software components of the radio that were produced before the OE's development for SCA compliancy began.

4.6.2.99 OE_TC_101 - Hardware Critical Interfaces

This test case verifies OE0633. The objective of this test is to verify that all hardware critical interfaces are defined in Interface Control Documents (OE0633). The definition of hardware critical interfaces is described to be those interfaces at the physical boundary of a replaceable device that are required for the operation and maintenance of the device.

4.6.2.100 OE_TC_102 - Base Application Interfaces

This test case verifies OE0703. The objective of this test is to verify that the Base Application Interfaces (Ports, LifeCycle, TestableObject, PropertySet, PortSupplier, Resource or ResourceFactory) follow the SCA v2.2.2 IDL specification presented in Appendix C [Reference 3].

4.6.2.101 OE_TC_103 - FileSystem :: list raises InvalidFileName

This test case verifies OE0534 and OE0598. The objective of this test case is to verify that the *remove* operation raises the CF::*FileException* when a file-related error occurs. This test case is to verify that the error number value in the exception is of the type CF::ErrorNumberType.

4.6.2.102 OE_TC_104 - FileSystem :: list raises FileException

This test case verifies the requirement OE0546 and OE0598. The objective of this test is to verify that the *list* operation raises the CF::FileException when a file-related error occurs (OE0546). The error number of the exception must be one of the CF::ErrorNumberType values.

4.6.2.103 OE_TC_105 - FileSystem :: remove raises FileException

This test case verifies OE0534 and OE0598. The objective of this test case is to verify that the *remove* operation raises the CF::*FileException* when a file-related error occurs. This test case is to verify that the error number value in the exception is of the type CF::ErrorNumberType.

4.6.2.104 OE_TC_106 - FileSystem :: copy

This test case verifies OE0535 and OE0734. The test is to verify that the *copy* operation found in the FileSystem and FileManager copies the file identified by the input parameter *sourceFileName* to the destination file identified by the input parameter *destinationFileName*. It also verifies that should the destination file exist and differ from the source file, the *copy* operation will overwrite it with the source file. The files mentioned in these requirements are plain files.

4.6.2.105 OE_TC_107 - FileSystem :: create

This test case verifies requirement OE0547. The objective of this test is to verify that a new file is created via the *create* operation using the input *fileName* parameter.

4.6.2.106 OE_TC_108 - Application :: releaseObject releases all objects

This test case verifies OE0137. The test will verify that the *releaseObject* operation of an instantiation of the *Application* class releases all object references to the components, which make up an application. In this test case, *Application* will refer to an instantiation of the *Application* class.

4.6.2.107 OE_TC_109 - FileSystem's CREATED_TIME_ID property

This test case verifies OE0529. The objective of this test is to verify that the CREATED_TIME_ID property is implemented and that the data type is correct.

4.6.2.108 OE_TC_110 - FileSystem's MODIFIED_TIME_ID property

This test case verifies OE0530. The objective of this test is to verify that the MODIFIED_TIME_ID property is implemented and that the data type is correct.

4.6.2.109 OE_TC_111 - FileSystem's LAST_ACCESS_TIME_ID property

This test case verifies OE0531. The objective of this test is to verify that the LAST_ACCESS_TIME_ID property is implemented and that the data type is correct.

4.6.2.110 OE_TC_112 - DeviceManager's execparam properties

This test case verifies OE0754, and OE0756. The objective of this test is to verify that the DeviceManager(s) correctly provides the *execparam* properties as a parameter to the *ExecutableDevice::execute* operation, when deploying services. The *execparam* should contain IDs and values of type string.

4.6.2.111 OE_TC_113 - DomainManager :: registerDevice

This test case verifies OE0719. The objective of this test is to verify that the DomainManager's registerDevice operation establishes connections that are pending. Pending connections are broken connections as a result from *unregisterDevice* operations. If the new device that is being registered meets the criteria of the needs of the pending connection then the new device will complete the connection. A pending connection can exist when a new device is registered and its connection counterpart is not yet existent in the domain.

4.6.2.112 OE_TC_114 - FileSystem :: mkdir

This test case verifies requirement OE0560. The objective of this test is to verify that the *mkdir* operation creates a file system directory based on the *directoryName* parameter.

4.6.2.113 OE_TC_115 - FileSystem :: rmdir

This test case verifies requirement OE0564. The objective of this test is to verify that the *rmdir* operation removes the directory identified by the input parameter *directoryName*. Assumption is that the directory identified by the input *directoryName* parameter is empty. This test case does not check if the directory has any files and/or directories.

4.6.2.114 OE_TC_116 - Framework Control Interfaces

This test case verifies OE0120. The objective of this test is to verify that the Framework Control Interfaces are implemented using the CF.idl as presented in Appendix C. The CF.idl implements the CF module.

4.6.2.115 OE_TC_117 - DTD files

This test case verifies requirement OE0589. The objective of this test is to verify that the DTD (Document Type Definitions) files are installed in the domain and have the "***.dtd***" filename extensions. The document type declaration (!DOCTYPE) specifies the DTD for the document. The names of the DTD files installed in the domain are found in SCA v2.2.2 Attachment 1 to Appendix D.

4.6.2.116 OE_TC_118 - Device :: allocateCapacity's acceptable properties

This test case verifies OE0730. The objective of this test is to verify that only properties described in the component's SPD with a *kindtype* of *allocation* and an *action* element of *external* are accepted by the *allocateCapacity* operation.

4.6.2.117 OE_TC_119 - Domain Profile files

This test case verifies OE0613. The objective of this test is to verify that the OE is described by the XML Domain Profile files properly. It will also build lists for use in other test cases.

4.6.2.118 OE_TC_121 - ApplicationFactory :: create does property comparisons

This test case verifies OE0707. The objective of this test is to verify that property comparisons are performed in accordance with the data in the Domain Profile files.

4.6.2.119 OE_TC_122 - DomainManager :: uninstallApplication raises ApplicationUninstallationError

This test case verifies requirement OE0301, OE0309 and OE0203. The objective of this test is to verify that the *uninstallApplication* operation provided by the domain manager writes a FAILURE_ALARM log record to a domain manager's log and raises the ApplicationUninstallationError exception when an internal error prohibits the successful uninstallation of the application. The exception should also be accompanied with a CF::ErrorNumberType value (OE0203).

4.6.2.120 OE_TC_124 - DomainManager :: registerDevice raises RegisterError

This test case verifies OE0248, OE0258 and OE0201. The objective of this test is to verify that the *RegisterError* is raised when an internal error causes an unsuccessful registration during the *DomainManager::registerDevice* operation, and verify that the error number value in the exception is of the type *ErrorNumberType*. This test case also verifies OE0248, which means a FAILURE_ALARM record is written to the domain manager log when the *RegisterError* exception is raised.

4.6.2.121 OE_TC_125 - ApplicationFactory:: create raises CreateApplicationError

This test case verifies OE0197 and OE0152. The objective of this test is to verify that the ApplicationFactory *create* operation raises the *CreateApplicationError* exception when a *create* requests is valid but the application cannot be instantiated due to internal processing error(s). Furthermore, the test case verifies that the exception contains a message and an error number of the type CF::ErrorNumberType.

4.6.2.122 OE_TC_126 - DomainManager :: registerDeviceManager raises RegisterError

This test case verifies OE0241 and OE0201. The objective of this test is to verify that the *RegisterError* is raised when an internal error exists causing an unsuccessful registration during the *registerDeviceManager* operation, and to verify that the error number value in the exception is of the type ErrorNumberType. This test also verifies that a FAILURE_ALARM log record is written to a domain manager's log when the *RegisterError* exception is raised.

4.6.2.123 OE_TC_127 - DomainManager :: registerService raises RegisterError

This test case verifies OE0317, OE0326 and OE0201. The objective of this test is to verify that when an internal error exists which causes an unsuccessful registration during the *registerService* operation, the *RegisterError* exception is raised and the error number value in the exception is of the type *ErrorNumberType*.

4.6.2.124 OE_TC_128 - DomainManager :: unregisterDeviceManager raises UnregisterError

This test case verifies OE0278, OE0285 and OE0202. The objective of this test is to verify that when an internal error causes an unsuccessful unregistration (1) the unregisterDeviceManager operation writes a FAILURE_ALARM log record to a domain manager's log, (2) the unregisterDeviceManager operation raises the UnregisterError exception, (3) there is an error number as part of the raised exception, which is of the type CF::ErrorNumberType.

4.6.2.125 OE_TC_129 - DomainManager :: unregisterDevice raises UnregisterError

This test case verifies OE0297, OE0202 and OE0290. The objective of this test is to verify that, when an internal error causes an unsuccessful unregistration, (1) the *unregisterDevice* operation raises the *UnregisterError* exception (2) there is an appropriate error number that is of the type CF::ErrorNumberType, and (3) a FAILURE_ALARM log record is sent to the domain manager's log.

4.6.2.126 OE_TC_130 - LoadableDevice :: load raises LoadFail

This test case verifies OE0424 and OE0432. The objective of this test is to verify that the CF::LoadFail exception occurs when a *load* operation for a LoadableDevice fails. The CF::LoadFail exception indicates that the *load* operation failed due to device dependent reasons. The CF::LoadFail exception indicates that an error occurred during an attempt to load the device. OE0424 verifies that the exception includes an error number of the type ErrorNumberType.

4.6.2.127 OE_TC_131 - AEP mandatory functions

This test case verifies OE0657, OE0658, OE0659, OE0660, OE0663, OE0664, OE0665, OE0667, OE0668, OE0669, OE0670, OE0671, OE0672, OE0673, OE0762, OE0763, OE0764, OE0765, OE0767, OE0768, OE0769, and OE0770. The objective of this test is to verify that the OE supplies the functions designated as mandatory in Appendix B of the SCA [Reference 9] and to verify that the POSIX.1 options that are mandatory are provided.

4.6.2.128 OE_TC_132 - DomainManager :: installApplication raises InvalidFileName

This test case verifies OE0269, OE0270 and OE0599. The objective of this test is to verify that the DomainManager *installApplication* operation raises the *InvalidFileName* exception when the input SAD file or any of the SAD's referenced filenames do not exist in the file system identified by the absolute path of the input *profileFileName* parameter. Also, verify that DomainManager *installApplication* operation logs a FAILURE_ALARM log record to the DomainManager's log with an appropriate detailed message containing the filename that caused the *InvalidFileName* to be raised. Finally, verify the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.129 OE_TC_133 - LoadableDevice :: load raises InvalidFileName

This test case verifies OE0431 and OE0599. The objective of this test is to verify that the LoadableDevice *load* operation raises the *InvalidFileName* exception when the file designated by the input filename parameter cannot be found. Furthermore, verify the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.130 OE_TC_134 - LoadableDevice :: unload raises InvalidFileName

This test case verifies OE0436 and OE0599. The objective of this test is to verify that the LoadableDevice *unload* operation raises the *InvalidFileName* exception when the file designated by the input filename parameter cannot be found. Furthermore, verify the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.131 OE_TC_135 - ExecutableDevice :: execute raises InvalidFileName

This test case verifies OE0452 and OE0599. The objective of this test is to (1) verify that the *execute* operation raises the *InvalidFileName* exception when the file name indicated by the input

name parameter does not exist for the device, (2) verify that the error number in the exception is of the type CF::ErrorNumberType.

4.6.2.132 OE_TC_136 - FileSystem :: remove raises InvalidFileName

This test case verifies OE0533 and OE0599. The objective of this test is to verify that the *remove* operation raises the *CF::InvalidFileName* exception when the input filename parameter is invalid or not an absolute path name. This test case also verifies that the exception must include an error number of *CF::ErrorNumberType*.

4.6.2.133 OE_TC_137 - FileSystem :: copy raises InvalidFileName when inputs are invalid

This test case verifies OE0537 and OE0599. The objective of this test is to (1) verify that the *copy* operation raises the *CF::InvalidFileName* exception when the *sourceFileName* or *destinationFileName* input parameters are not a valid absolute pathname (2) verify that the error number of the exception is of the type, CF::ErrorNumberType.

4.6.2.134 OE_TC_138 - FileSystems :: exists

This test case verifies the requirements, OE0538 and OE0539. The objective of this test is to verify that the *exists* operation checks that the file represented by the filename parameter exists in the directory structure and returns the correct value.

4.6.2.135 OE_TC_139 - FileSystem :: create raises InvalidFileName

This test case verifies OE0551 and OE0599. The objective of this test is to verify that the *create* operation raises the *CF::InvalidFileName* exception when the input fileName parameter is not a valid absolute pathname(OE0551) and to verify that the error number of the exception is of the type, CF::ErrorNumberType (OE0599).

4.6.2.136 OE_TC_140 - FileSystem :: open raises InvalidFileName

This test case verifies OE0559 and OE0599. The objective of this test is to (1) verify that the *open* operation raises the *CF::InvalidFileName* exception when the input fileName parameter is not a valid absolute pathname, (2) verify that the error number of the exception is of the type, CF::ErrorNumberType.

4.6.2.137 OE_TC_141 - FileSystem :: mkdir raises InvalidFileName

This test case verifies OE0563 and OE0599. The objective of this test case is to verify the *mkdir* operation for the specific situations

(1) raises the *CF::InvalidFileName* exception when the input parameter – *directoryName*, is not a valid directory name (OE0563),

(2) provides an error number with the exception of type, CF::ErrorNumberType (OE0599).

4.6.2.138 OE_TC_142 - FileSystem :: rmdir raises InvalidFileName

This test case verifies OE0566 and OE0599. The objective of this test is to verify that the FileSystem *rmdir* operation raises the *InvalidFileName* exception when the input directoryName parameter is not a valid path prefix. Furthermore, verify the exception contains an error number of the type CF::ErrorNumberType. In SCA v2.2.2, section 1.3.1.1, it states ‘A “path prefix” is a

pathname which refers to a directory and thus does not include the name of a plain file.' Thus, the entire input parameter directoryName is a "path prefix".

4.6.2.139 OE_TC_143 - FileManager :: mount raises InvalidFileName

This test case verifies OE0575 and OE0599. The objective of this test is to (1) verify that the *mount* operation raises the *InvalidFileName* exception when the input mount point does not conform to the file name syntax in section 3.1.3.4.2.1. , (2) verify that there is an appropriate error number of the type CF::ErrorNumberType as part of the raised exception.

4.6.2.140 OE_TC_144 - FileSystem :: copy raises InvalidFileName when inputs are the same

This test case verifies OE0735 and OE0599. The objective of this test is to verify that the *copy* operation throws the CF::InvalidFileName exception when the source and destination files are the same file and to verify the error number type accompanying the exception.

4.6.2.141 OE_TC_145 - ExecutableDevice :: terminate raises InvalidProcess

This test case verifies OE0458 and OE0438. The objective of this test is to:

(1) Verify that the *terminate* operation raises the *InvalidProcess* exception when the process Id does not exist for the device.
(2) Verify that the error number in the exception is of the type CF::ErrorNumberType.

4.6.2.142 OE_TC_146 - File :: write raises IOException

This test case verifies OE0519 with criteria OE0519-C111, and OE0507. The objective of this test is to (1) verify that the *write* operation raises the *IOException* exception when a write error occurs, (2) when a read_Only file is opened to write, this should cause a write error; therefore the File::write operation is expected to raise the IOException exception, (3) verify that the error number in the exception is of the type CF::ErrorNumberType.

4.6.2.143 OE_TC_147 - File :: sizeOf raises FileException

This test case verifies OE0521 and OE0598. The objective of this test is to (1) verify that the *FileException* is raised when a file related error occurs during the *sizeOf* operation, and (2) verify that there is an appropriate error number of *CF::ErrorNumberType* as part of the exception.

4.6.2.144 OE_TC_148 - FileSystem :: copy raises FileException

This test case verifies OE0536 and OE0598. The objective of this test is to (1) verify that the *FileException* is raised when a file related error occurs during the *copy* operation, (2) verify that the error number value in the exception is of the type *ErrorNumberType*.

4.6.2.145 OE_TC_149 - FileSystem :: create raises FileException

This test case verifies OE0550 and OE0598. The objective of this test is three-fold (1) verify that the *FileException* is raised when the file already exists or another file error occurs during the File *create* operation, (2) verify that the error number value in the exception is of the type *ErrorNumberType*.

4.6.2.146 OE_TC_150 - FileSystem :: open raises FileException

This test case verifies OE0558 and OE0598. The objective of this test is to verify that the FileSystem and FileManager *open* operation raises the CF *FileException* exception if the file does

not exist or another file error occurred. Furthermore, verify the exception contains an error number of the type CF::ErrorNumberType.

4.6.2.147 OE_TC_151 - FileSystem :: mkdir raises FileException

This test case verifies OE0562 and OE0598. The objective of this test is to (1) verify that the *mkdir* operation raises the *FileException* exception if the directory indicated by the input directoryName parameter already exists or if a file-related error occurred during the operation, (2) verify that there is an appropriate error number of type CF::ErrorNumberType as part of the raised exception.

4.6.2.148 OE_TC_152 - FileSystem :: rmdir raises FileException

This test case verifies OE0565 and OE0598. The objective of this test is to (1) verify that the *FileException* is raised during the *rmdir* operation when the directory identified by the input *directoryName* parameter does not exist, the directory contains files, or an error occurs which prohibits the directory from being deleted, (2) verify that the error number value in the exception is of the type *ErrorNumberType*.

4.6.2.149 OE_TC_153 - LoadableDevice :: load

This test case verifies OE0731. The objective of this test is to verify that the result of multiple loads of the same input *fileName* parameter does not raise an exception.

4.6.2.150 OE_TC_154 - DomainManager :: registerDeviceManager sends event to ODM

This test case verifies OE0234 and OE0234-C041 through OE0234-C045. The objective of this test is to verify that the *registerDeviceManager* operation sends a DomainManagementObjectAddedEventType to the Outgoing Domain Management event channel upon successful registration of a device manager. In addition, this test will verify that the DomainManagementObjectAddedEventType event sent by the *registerDeviceManager* operation contains the appropriate data for *producerId*, *sourceId*, *sourceName*, *sourceIOR* and *sourceCategory*.

4.6.2.151 OE_TC_155 - Naming Context's execute parameter

This test case verifies requirement OE0750. There are 3 parts to verify this requirement, of which all 3 parts must be verified and validated to pass this requirement. The first part is to verify the execute parameter for the Naming Context IOR is a CF::Properties type. The second part is the Naming Context IOR is of CF::Properties Type defined with an id element set to "NAMING_CONTEXT_IOR". Lastly, a value element set to the stringified IOR of the naming context.

4.6.2.152 OE_TC_156 - Naming Context creation

This test case verifies SCA v2.2.2 Operating Environment Requirement number OE0720 and OE0745. The objective of this test case is to verify that the DomainManager:

(1) Creates a naming context, followed by

(2) It creating a "name binding" to the created naming context in step 1.

4.6.2.153 OE_TC_157 - DomainManager :: registerDeviceManager establishes connections

This test case verifies requirement OE0226 and criterion OE0226-C040. The objective of this test is to verify that the *registerDeviceManager* operation establishes connections for the device manager in the input deviceMgr parameter. The DeviceManager's DCD file specifies the connections for the DeviceManager. Also, verify that if the connection is not currently possible then the connection is left unconnected with no error.

4.6.2.154 OE_TC_158 - DomainManager :: unregisterDeviceManager

This test case verifies OE0279 and OE0279-C059 through OE0279-C062. The objective of this test is to verify that the *unregisterDeviceManager* operation sends a DomainManagementObjectRemovedEventType to the Outgoing Domain Management event channel upon successful un-registration of a device manager. In addition, this test will verify that the DomainManagementObjectRemovedEventType event sent by the *unregisterDeviceManager* operation contains the appropriate data for *producerId*, *sourceId*, *sourceName* and *sourceCategory*.

4.6.2.155 OE_TC_159 - FileSystem :: exists raises InvalidFileName

This test case verifies OE0540 and OE0599. The objective of this test is to (1) verify that the *exists* operation raises the *CF::InvalidFileName* exception when the input *fileName* parameter is not a valid absolute pathname, (2) verify that the error number of the exception is of the type, CF::ErrorNumberType.

4.6.2.156 OE_TC_160 - Name Binding's execute parameter

This test case verifies OE0751. The objective of this test is to verify that the Name Binding execute parameter is a CF::*Properties* type with an id element set to "NAME_BINDING" and a value element set to a string in the format of "ComponentName_UniqueIdentifier".

4.6.2.157 OE_TC_161 - Component Identifier's execute parameter

This test case verifies requirement OE0752. The objective of this test is to verify that the execute parameter for the Component Identifier is a CF Properties type with an id element set to "COMPONENT_IDENTIFIER" and a value element set to a string in the format of "Component_Instantiation_Identifier: Application_Name".

4.6.2.158 OE_TC_162 - ApplicationFactory :: create deallocates capacity on devices

This test case verifies OE0708. The objective of this test is to verify that the *create* operation shall deallocate any capacity allocations on devices that do not satisfy the application components allocation requirements or that are not utilized due to an unsuccessful application creation.

4.6.2.159 OE_TC_164 - ApplicationFactory :: create establishes connections for named applications

This test case verifies OE0710. The objective of this test is to verify that Operating Environment(OE) provides the functionality in the CF::ApplicationFactory *create* operation to establish all of the connections for an application. The connections for the application being instantiated are specified by the application's SAD file, specified in the domainfinder element.

4.6.2.160 OE_TC_165 - ApplicationFactory :: create defines an order for initialization

This test case verifies OE0174. The objective of this test is to verify the order in which the *create* operation will perform some of its tasks. Specifically it requires all resources to be initialized, then all connections to be made, and finally, the component to be configured. The assembly controller is always required because all external requests of runTest, start, stop, configure, and query are delegated through it to the proper application.

4.6.2.161 OE_TC_166 - DomainManager :: registerDevice verifies input parameters

This test case verifies OE0715. The objective of this test is to verify that the *registerDevice* operation validates its parameters (registeringDevice and registeredDeviceMgr) are not nil CORBA object references.

4.6.2.162 OE_TC_167 - DomainManager :: registerDevice returns without error if device exists

This test case verifies OE0716. The objective of this test is to verify that the *registerDevice* operation will not throw an exception and not register a new device, when that device is a duplicate and the reference refers to an existing object. The *registerDevice* operation registers a Device for a specific device manager with the DomainManager.

4.6.2.163 OE_TC_168 - DomainManager :: registerDevice registers if device doesn't exist

This test case verifies OE0717 and OE0718. The registerDevice operation should implement a check for registering a device with a previously registered identifier. The registerDevice operation should register the device indicated if the identifier attribute has not been previously registered or if it has been previously registered and the reference is to a nonexistent object. The test also verifies that an ADMINISTRATIVE_EVENT log record is written to the log when the reference to the registered device refers to a nonexistent object.

4.6.2.164 OE_TC_189 - Application Delegates Implementation of Resource operations

This test case verifies OE0127, OE0128, and OE0129. The objective of this test is to ensure that the Application interface properly passes requests to the assembly controller for the runTest, start, stop, configure, and query operations. It also verifies that exceptions are properly passed back through the Application interface from the runTest, start, stop, configure, and query operations. This test also ensures that exceptions from the Application interfaces initialize operation are not propagated to the application's components.

4.6.2.165 OE_TC_218 - Device Attributes

This test case verifies OE0395, OE0401, OE0403, OE0404, and OE0729. The objective of this test is to verify a device's operationalState, softwareProfile, label and compositeDevice read-only attributes. With regard to requirement OE0401, a discussion of absolute and relative pathnames may be found in the SCA section 1.3.1.1 on page 1-2.

4.6.2.166 OE_TC_221 - ExecutableDevice Types

This test verifies requirements OE0442 and OE0443. The objective of this test case is to confirm that the *execute* method of the ExecutableDevice interface enforces the types of the STACK_SIZE and PRIORITY contained in the "options" parameter.

4.6.2.167 OE_TC_222 - ExecutableDevice :: execute raises exceptions

This test case verifies OE0450, OE0451, OE0453, OE0454, OE0455 and OE0444. The objective of this test is to verify that the Executable Device *execute* operation raises various exceptions based on the circumstances of a failed execution. It raises the *InvalidState* exception when at the beginning of the execution, the device's adminState attribute is LOCKED or SHUTTING_DOWN or its operationalState attribute is DISABLED. It raises the *InvalidParameters* exception when the input parameter ID or value attributes are not valid strings. It raises the *InvalidOptions* exception when the input options parameter does not comply with sections that describe STACK_SIZE_ID and PRIORITY_ID. It raises the *InvalidFunction* exception when the function indicated by the input name parameter does not exist for the device. Finally, it raises the *ExecuteFail* exception when the operating system "execute" function for the device is not successful. Furthermore, the test case verifies that the *ExecuteFail* exception contains an error number of the type CF::ErrorNumberType.

4.6.2.168 OE_TC_224 - DeviceManager Attributes

This test case verifies OE0468, OE0469, OE0470, OE0471, and OE0472. The objective of this test is to verify a deviceManager interface's label, fileSys, deviceConfigurationProfile, registeredDevices and registeredServices read-only attributes.

4.6.2.169 OE_TC_227 - ExecutableDevice :: execute

This test case verifies OE0445, OE0446, OE0447 and OE0448. The objective of this test is to verify that the Executable Device *execute* operation executes the entity identified by the *id* parameter using the given *parameters* and *options* parameters. This test is to verify the proper parsing of its *parameters* parameter and its *options* parameter. It also will verify that should the *options* parameters include STACK_SIZE_ID and PRIORITY_ID, they will be used to set the OS' process/thread stack size and priority.

4.6.2.170 OE_TC_229 - Device :: allocateCapacity raises exceptions

This test case verifies OE0409 and OE0410. The objective of this test case is to verify that the *allocateCapacity* operation shall raise the InvalidCapacity exception, when the input capacities parameter contains invalid properties or when attributes of those CF::Properties contain an unknown id or a value of the wrong data type. Furthermore, the objective of this test case is to verify that the allocateCapacity operation shall raise the InvalidState exception, when the Device's adminState is not UNLOCKED or operationalState is DISABLED.

4.6.2.171 OE_TC_230 - Device :: deallocateCapacity raises InvalidState

This test case verifies OE0417. The objective of this test case is to verify that the deallocateCapacity operation raises the InvalidState exception, when the device's adminState is LOCKED or operationalState is DISABLED.

4.6.2.172 OE_TC_233 - Application Attributes

This test case verifies OE0121, OE0122, OE0123, OE0124, OE0125, and OE0126. The test objective is to verify the attributes profile, name, componentNamingContexts, componentProcessIds, componentDevices and componentImplementations of an implemented Application. Attributes will be verified for type, content, and availability.

4.6.2.173 OE_TC_236 - ApplicationFactory Attributes

This test case verifies OE0153, OE0154, OE0155 and OE0156. The objective of this test is to verify the ApplicationFactory attributes name, identifier and softwareProfile are correctly defined. The test verifies that the name attribute is identical to the softwareassembly element name attribute of the application's SAD. It verifies that the softwareProfile attribute contains a profile element with a file reference to the application's SAD. It also verifies that the identifier attribute contains the unique identifier for an ApplicationFactory instance and that it be identical to the softwareassembly element id attribute of the application factory's SAD.

4.6.2.174 OE_TC_265 - File Attributes

This test verifies requirements OE0509 and OE0510. The objective of this test case is to confirm that the attributes of the File interface are implemented correctly. First, verify the fileName attribute contains the pathname used as the input fileName parameter of the FileSystem::create operation when the file was created. Secondly, confirm the readonly filePointer attribute contains the current file position.

4.6.2.175 OE_TC_275 - FileSystem :: query Input Parameter

This test case verifies OE0567, OE568 and OE569. The objective of this test is to verify that the *query* operation returns file system information based on the given fileSystemProperties' ID. It verifies that the *query* operation recognizes the ID values of SIZE and AVAILABLE SPACE and returns the corresponding information. Finally, it verifies that when an input file system property is not recognized, then an *UnknownFileSystemProperties* exception is raised.

4.6.2.176 OE_TC_276 - FileManager :: mount

This test case verifies OE0573, OE0574, OE0576 and OE0577. The objective of this test case is to verify that the *mount* operation validates the input mountPoint parameter, and verifies that it does not already exist. Furthermore, the test case verifies that the *mount* operation raises an exception if the mount point does already exist, and it raises an exception if the input file system object reference is null. Finally, the test case verifies that the *mount* operation associates the specified file system with the mount point referenced by the input parameter.

4.6.2.177 OE_TC_283 - FileSystem create Responsibilities

This test case verifies OE0475, OE0476, OE0476-C109 and OE0760. The objective of this test case is to verify that the deviceManager creates a FileSystem component for each OS file system. It also verifies that

- the deviceManager is responsible for mounting multiple file systems if present to a FileManager component,
- the mount points used for the created files systems match those found in the DCD file, and
- each mounted file system has a unique name within the device manager.

4.6.2.178 OE_TC_285 - DeviceManager startup process

This test case verifies OE0481, OE0482, and OE0483. The objective of this test case is to verify that the Device Manager performs certain parts of its start-up tasks as specified. The tasks to be verified are that the Device Manager

- parses the DCD and SPD files
- uses the *componentinstantiation* element's SPD implementation code's *stacksize* and *priority* elements, when specified, for the *execute* operation options parameters, (corresponds to step 6 of the SCA figure 3-19)
- initializes and configures logical devices that are started by the Device Manager after they have successfully registered with the DeviceManager, and (corresponds to step 9 of the SCA figure 3-19)
- configures a DCD's componentinstantiation element provided the componentinstantiation element has "configure" readwrite or writeonly properties with values. (corresponds to step 10 of the SCA figure 3-19)

4.6.2.179 OE_TC_290 - Networking AEP

This test case verifies OE0819 and OE0820. The objective of this test is to verify that the OE supplies the functions designated as mandatory in SCA Networking Application Environment Profile. According to the direction of JTNC Standards, it is not considered as failure when the OE cannot provide any or all of the mandatory functions listed in Table 1-2 and 1-3 in the SCA Networking Application Environment Profile.

4.6.3 AEP Amended Requirements Test Case Objectives (Appendix C)

4.6.3.1 OE_TC_170 - AEP Amended Applications have no abnormal termination

This test case verifies OE0798. The objective of this test is to verify that if an application conforms to the AEP, the process will not abnormally terminate. The default action for all of these signals is to cause the process to terminate. Handler functions that terminate the program are typically used to cause orderly cleanup or recovery from program error signals and interactive interrupts.

4.6.3.2 OE_TC_172 - AEP Amended mandatory functions

This test case verifies OE0790, OE0794, OE0795, OE0796, OE0797, OE0811, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0812, OE0813, OE0814, OE0815, OE0816, OE0818, and OE0817. The objective of this test is to verify that the OE defines the functions designated as mandatory in Appendix B of the SCA v2.2.2A [Reference 2] and to verify that the POSIX.1 options that are mandatory are provided.

4.6.3.3 OE_TC_173 - AEP Amended file mode creation masks

This test case verifies OE0803. The objective of this test is to verify that the file mode creation mask for any object created is at a minimum S-IRWXU. This mask says that the creator, i.e., owner, will have read, write, and execute rights to the object created. The file mode creation mask is used for the creation of directories and files.

4.6.3.4 OE_TC_175 - AEP Amended Standard C Library header files

This test case verifies OE0791. The objective of this test is to verify that the Core Framework V the Standard C Library header files designated as mandatory in Appendix B [Reference 2] of the SCA v2.2.2A.

4.6.4 SCA Extension Requirements Test Case Objectives (Appendix D)

4.6.4.1 OE_TC_176 - DomainManager :: registerService registers non-SCA services

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.

4.6.4.2 OE_TC_177 - DomainManager :: unregisterService removes non-SCA services

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 performs the service using either the fully qualified name or a simple name using the "identifier" portion only.

4.6.4.3 OE_TC_178 - ApplicationFactory :: create recognizes DEPLOYMENT_CHANNEL options

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.

4.6.4.4 OE_TC_179 - ApplicationFactory :: create recognizes DEFAULT deployment options

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

4.6.4.5 OE_TC_180 - ApplicationFactory :: create with "servicetype" connections to a non-SCA service

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.

4.6.4.6 OE_TC_181 - ApplicationFactory :: create raises InvalidInitConfiguration

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.

4.6.4.7 OE_TC_182 - ApplicationFactory :: create raises CreateApplicationError when not able to allocate applications properly

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 [Reference 10] 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.

4.6.4.8 OE_TC_183 - ApplicationFactory :: create raises CreateApplicationError when a "servicetype" connection cannot be established

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 [Reference 10] 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.

4.6.4.9 OE_TC_184 - DeviceManager's execparam properties

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.

4.6.4.10 OE_TC_185 - Domain Profile

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 [Reference 10] 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.

5 Requirements Traceability Matrix

The Requirements Traceability Matrix ensures completeness by associating the test case number with the SCA requirement tag, criteria tag, and requirement text.

Other information includes the section and requirement context found in the referenced documents.

The criteria tag denotes the behaviors that are associated with specific requirements. These behaviors or criterions are not 'shall' requirements, but aids in interpreting the intent of a requirement. For identification, the criterion uses the "C" character and three digits.

Table 3: Column Headers for Requirements Traceability Matrix

Column Header	Meaning	Example
Requirement Tag	Identifier of the SCA v2.2.2 requirement	OE0700
Criterion Tag	Indicates that there is one or more criterion associated with the requirement. The text of criterions may be found in their associated Test Cases in Appendix B	C182
Test Case Number	Comprised of "OE" (Operating Environment) plus "TC (test case), and "xxx" (unique 3-digit number)	OE_TC_005
Requirement Text	Text of the requirement	This operation returns the maximum capacity of the storage area.
Section	Section Number of the OMG Lightweight Log Service or SCA v2.2.2 Specification document [Reference 7]	3.3.1.1
App Volume	The Appendix letter or if in Appendix B the volume number that the test case may be found in. There are 5 volumes to Appendix B	2 or App D

The Requirements Traceability Matrix is sorted on the Requirement Tag column. That is, Table 4 is ordered on the requirements numbers whose test cases appear in this document. Criterion numbers are a secondary sorting value for the table.

Table 4: Requirements Traceability Matrix

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0001		OE_TC_059	The OE shall provide the functions and options designated as mandatory by the AEP defined in Appendix B.	3.1.1	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0002		OE_TC_060	The OE and related file systems shall support a filename length of 40 characters and a pathname length of 1024 characters.	3.1.1	4
OE0003		OE_TC_061	The OE shall include middleware that, at a minimum, provides the services and capabilities of minimumCORBA as specified by the OMG Document in reference [Reference 7].	3.1.2	5
OE0007		OE_TC_001	The "kind" element of each NameComponent shall be "" (null string).	3.1.2.1	5
OE0011		OE_TC_002	A log producer shall only output log records that contain an enabled CosLwLog::LogLevel value.	3.1.2.2.1	5
OE0012		OE_TC_002	Log producers shall use their component identifier attribute in the producerId field of the CosLwLog::ProducerLogRecord.	3.1.2.2.1	5
OE0013		OE_TC_002	Log producers and CF components that are required by this specification to write log records shall operate normally in the absence of a log service or in the case where the connections to a log are nil or an invalid reference.	3.1.2.2.1	5
OE0063		OE_TC_063	A component (e.g., Resource, DomainManager, etc.) that consumes events shall implement the CosEventComm PushConsumer interface.	3.1.2.3.1	5
OE0064		OE_TC_063	A component (e.g., Resource, Device, DomainManager, etc.) that produces events shall implement the CosEventComm PushSupplier interface and use the CosEventComm PushConsumer interface for generating the events.	3.1.2.3.1	5
OE0065		OE_TC_063	A producer component shall not forward or raise any exceptions when the connection to a CosEventComm PushConsumer is a nil or invalid reference.	3.1.2.3.1	5
OE0066		OE_TC_003	The Incoming Domain Management Channel name shall be "IDM_Channel".	3.1.2.3.1	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0067		OE_TC_003	The Outgoing Domain Management Channel name shall be “ODM_Channel”.	3.1.2.3.1	5
OE0069	C002	OE_TC_020	The connectPort operation shall make a connection to the component identified by its input parameters.	3.1.3.1.1.5.1.3	1
OE0070	C004	OE_TC_020	The connectPort operation shall raise the InvalidPort exception when the input connection parameter is an invalid connection for this port.	3.1.3.1.1.5.1.5	1
OE0071		OE_TC_032	The connectPort operation shall raise the OccupiedPort exception when unable to accept the connections because the port is already fully occupied.	3.1.3.1.1.5.1.5	1
OE0072		OE_TC_035	The disconnectPort operation shall break the connection to the component identified by the input connectionId parameter.	3.1.3.1.1.5.2.3	1
OE0073	C003	OE_TC_035	The disconnectPort operation shall raise the InvalidPort exception when the input connectionId parameter is not a known connection to the Port component.	3.1.3.1.1.5.2.5	1
OE0074		OE_TC_037	The initialize operation shall raise an InitializeError exception when an initialization error occurs.	3.1.3.1.2.5.1.5	1
OE0075		OE_TC_038	The releaseObject operation shall release all internal memory allocated by the component during the life of the component.	3.1.3.1.2.5.2.3	1
OE0078		OE_TC_039	The releaseObject operation shall raise a ReleaseError exception when a release error occurs.	3.1.3.1.2.5.2.5	1
OE0079		OE_TC_041	The runTest operation shall use the input testId parameter to determine which of its predefined test implementations should be performed.	3.1.3.1.3.5.1.3	1
OE0080		OE_TC_042	The id/value pair(s) of the testValues parameter shall be used to provide additional information to the implementation-specific test to be run.	3.1.3.1.3.5.1.3	1

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0081		OE_TC_043	The runTest operation shall return the result(s) of the test in the testValues parameter.	3.1.3.1.3.5.1.3	1
OE0082		OE_TC_072	Valid testId(s) and both input and output testValues (properties) for the runTest operation shall at a minimum be the test properties defined in the properties test element of the component's Properties Descriptor (refer to Appendix D Domain Profile [Reference 4]).	3.1.3.1.3.5.1.3	1
OE0083		OE_TC_045	All testValues parameter properties (i.e., test properties defined in the propertyfile(s) referenced in the component's SPD) shall be validated.	3.1.3.1.3.5.1.3	1
OE0084		OE_TC_046	The runTest operation shall not execute any testing when the input testId or any of the input testValues are not known by the component or are out of range.	3.1.3.1.3.5.1.3	1
OE0087		OE_TC_093	The exception parameter invalidProperties shall contain the invalid testValues properties id(s) that are not known by the component or the value(s) are out of range.	3.1.3.1.3.5.1.5	1
OE0091		OE_TC_047	The configure operation shall assign values to the properties as indicated in the input configProperties parameter.	3.1.3.1.5.5.1.3	1
OE0092		OE_TC_048	Valid properties for the configure operation shall at a minimum be the configure readwrite and writeonly properties referenced in the component's SPD.	3.1.3.1.5.5.1.3	1
OE0093		OE_TC_049	The configure operation shall raise a PartialConfiguration exception when some configuration properties were successfully set and some configuration properties were not successfully set.	3.1.3.1.5.5.1.5	1
OE0099		OE_TC_050	The errorNumber parameter shall indicate a CF ErrorNumberType value.	3.1.3.1.6.3.1	1

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0100		OE_TC_064	The errorNumber parameter shall indicate a CF ErrorNumberType value.	3.1.3.1.6.3.2	1
OE0102		OE_TC_071	The stop operation shall disable all current operations and put the resource in a non-operating condition.	3.1.3.1.6.5.2.3	1
OE0103		OE_TC_064	The stop operation shall raise the StopError exception if an error occurs while stopping the resource.	3.1.3.1.6.5.2.5	1
OE0105		OE_TC_050	The start operation shall raise the StartError exception if an error occurs while starting the resource.	3.1.3.1.6.5.1.5	1
OE0120		OE_TC_116	Framework Control Interfaces shall be implemented using the CF IDL presented in Appendix C.	3.1.3.2	2
OE0121		OE_TC_233	The readonly profile attribute shall contain a profile element (Profile Descriptor) with a file reference to the application's SAD file.	3.1.3.2.1.4.1	1
OE0122		OE_TC_233	This readonly name attribute shall contain the name of the created application.	3.1.3.2.1.4.2	1
OE0123		OE_TC_233	The componentNamingContexts attribute shall contain the list of components' Naming Service Context within the application for those components using CORBA Naming Service.	3.1.3.2.1.4.3	1
OE0124		OE_TC_233	The componentProcessIds attribute shall contain the list of components' process IDs within the Application for components that are executing on a device.	3.1.3.2.1.4.4	1
OE0125		OE_TC_233	The componentDevices attribute shall contain a list of devices, which each component either uses, is loaded on or is executed on.	3.1.3.2.1.4.5	1
OE0126		OE_TC_233	The componentImplementations attribute shall contain the list of components' SPD implementation IDs within the application for those components created.	3.1.3.2.1.4.6	1
OE0137		OE_TC_108	The application shall release all object references to the components making up the application.	3.1.3.2.1.6.1.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0152		OE_TC_125, OE_TC_182, OE_TC_183	The error number shall indicate a CF ErrorNumberType value.	3.1.3.2.2.3.2	2, App D
OE0153		OE_TC_236	The name attribute shall be identical to the softwareassembly element name attribute of the application's Software Assembly Descriptor file.	3.1.3.2.2.4.1	2
OE0154		OE_TC_236	The readonly softwareProfile attribute shall contain a profile element (Profile Descriptor) with a file reference to the application's SAD file.	3.1.3.2.2.4.2	2
OE0155		OE_TC_236	The readonly identifier attribute shall contain the unique identifier for an ApplicationFactory instance.	3.1.3.2.2.4.3	2
OE0156		OE_TC_236	The identifier shall be identical to the softwareassembly element id attribute of the application factory's Software Assembly Descriptor file.	3.1.3.2.2.4.3	2
OE0174		OE_TC_165	The create operation shall, in order, initialize all application resources, then establish connections for those resources, and finally configure the application component indicated by the assemblycontroller element in the SAD.	3.1.3.2.2.5.1.3	2
OE0184		OE_TC_074	The create operation shall create the specified event channel if the event channel does not exist.	3.1.3.2.2.5.1.3	2
OE0197		OE_TC_125	The create operation shall raise the CreateApplicationError exception when the create request is valid but the application cannot be successfully instantiated due to internal processing error(s).	3.1.3.2.2.5.1.5	2
OE0200		OE_TC_057	The error number shall indicate a CF ErrorNumberType value.	3.1.3.2.3.3.1	2
OE0201		OE_TC_124, OE_TC_126, OE_TC_127	The error number shall indicate a CF ErrorNumberType value.	3.1.3.2.3.3.7	2
OE0202		OE_TC_055, OE_TC_128, OE_TC_129	The error number shall indicate a CF ErrorNumberType value.	3.1.3.2.3.3.8	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0203		OE_TC_122	The error number shall indicate a CF ErrorNumberType value.	3.1.3.2.3.3.9	2
OE0217		OE_TC_054	The logs utilized by the DomainManager implementation shall be defined in the DMD.	3.1.3.2.3.5	2
OE0218		OE_TC_056	The domain manager shall begin to use a service specified in the DMD once the service is successfully registered with the domain manager via the registerDeviceManager or registerService operations.	3.1.3.2.3.5	2
OE0226	C040	OE_TC_157	The registerDeviceManager operation shall establish any connections for the device manager indicated by the input deviceMgr parameter, which are specified in the connections element of the device manager's Device Configuration Descriptor (DCD) file, that are possible with the current set of registered devices and services.	3.1.3.2.3.6.1.3	2
OE0233		OE_TC_126	The registerDeviceManager operation shall, upon unsuccessful device manager registration, write a FAILURE_ALARM log record to a domain manager's Log.	3.1.3.2.3.6.1.3	2
OE0234	C041, C042, C043, C044, C045	OE_TC_154	The registerDeviceManager operation shall send a DomainManagementObjectAdded EventType event to the Outgoing Domain Management event channel upon successful registration of a device manager.	3.1.3.2.3.6.1.3	2
OE0241		OE_TC_126	The registerDeviceManager operation shall raise the RegisterError exception when an internal error exists which causes an unsuccessful registration.	3.1.3.2.3.6.1.5	2
OE0248		OE_TC_124	The registerDevice operation shall write a FAILURE_ALARM log record to a domain manager log when the RegisterError exception is raised.	3.1.3.2.3.6.2.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0258		OE_TC_124	The registerDevice operation shall raise the RegisterError exception when an internal error exists which causes an unsuccessful registration.	3.1.3.2.3.6.2.5	2
OE0261		OE_TC_057	The installApplication operation shall, upon unsuccessful application installation, write a FAILURE_ALARM log record to a domain manager's log.	3.1.3.2.3.6.3.3	2
OE0268		OE_TC_057	The installApplication operation shall raise the ApplicationInstallationError exception when the installation of the application file(s) was not successfully completed.	3.1.3.2.3.6.3.5	2
OE0269		OE_TC_132	The installApplication operation shall raise the CF InvalidFileName exception when the input SAD file or any of the SAD's referenced filenames do not exist in the file system identified by the absolute path of the input profileFileName parameter.	3.1.3.2.3.6.3.5	2
OE0270		OE_TC_132	The installApplication operation shall log a FAILURE_ALARM log record to a domain manager's Log with a message consisting of "installApplication::invalid file is xxx", where "xxx" is the input or referenced filename, when the CF InvalidFileName exception occurs.	3.1.3.2.3.6.3.5	2
OE0274		OE_TC_053	The unregisterDeviceManager operation shall release all device(s) and service(s) associated with the device manager that is being unregistered.	3.1.3.2.3.6.4.3	2
OE0278		OE_TC_128	The unregisterDeviceManager operation shall, upon unsuccessful unregistration of a device manager, write a FAILURE_ALARM log record to a domain manager's log.	3.1.3.2.3.6.4.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0279	C059, C060, C061, C062	OE_TC_158	The unregisterDeviceManager operation shall send a DomainManagementObjectRemovedEventType event to the Outgoing Domain Management event channel, upon successful unregistration of a device manager.	3.1.3.2.3.6.4.3	2
OE0285		OE_TC_128	The unregisterDeviceManager operation shall raise the UnregisterError exception when an internal error exists which causes an unsuccessful unregistration.	3.1.3.2.3.6.4.5	2
OE0290		OE_TC_129	The unregisterDevice operation shall, upon unsuccessful unregistration of a device, write a FAILURE_ALARM log record to a domain manager's log.	3.1.3.2.3.6.5.3	2
OE0297		OE_TC_129	The unregisterDevice operation shall raise the UnregisterError exception when an internal error exists which causes an unsuccessful unregistration.	3.1.3.2.3.6.5.5	2
OE0301		OE_TC_122	The uninstallApplication operation shall, upon unsuccessful uninstall of an application, write a FAILURE_ALARM log record to a domain manager's log.	3.1.3.2.3.6.6.3	2
OE0309		OE_TC_122	The uninstallApplication operation shall raise the ApplicationUninstallationError exception when an internal error causes an unsuccessful uninstallation of the application.	3.1.3.2.3.6.6.5	2
OE0314		OE_TC_065	The registerService operation shall associate the input registeringService parameter with the input registeredDeviceMgr parameter in the domain manager, when the registeredDeviceMgr parameter indicates a device manager that is registered with the domain manager.	3.1.3.2.3.6.7.3	2
OE0317		OE_TC_127	The registerService operation shall, upon unsuccessful service registration, write a FAILURE_ALARM log record to a domain manager's log.	3.1.3.2.3.6.7.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0326		OE_TC_127	The registerService operation shall raise the RegisterError exception when an internal error exists which causes an unsuccessful registration.	3.1.3.2.3.6.7.5	2
OE0327		OE_TC_051	The unregisterService operation shall remove the unregisteringService entry specified by the input name parameter from the domain manager.	3.1.3.2.3.6.8.3	2
OE0328		OE_TC_052	The unregisterService operation shall release (client-side CORBA release) the unregisteringService from the domain manager.	3.1.3.2.3.6.8.3	2
OE0337		OE_TC_055	The unregisterService operation shall raise the UnregisterError exception when an internal error exists which causes an unsuccessful unregistration.	3.1.3.2.3.6.8.5	2
OE0345		OE_TC_079	The readonly usageState attribute shall contain the device's usage state (IDLE, ACTIVE, or BUSY).	3.1.3.3.1.4.1	3
OE0381	C082, C083, C084, C085, C086	OE_TC_079	The device shall send a StateChangeEventType event to the Incoming Domain Management event channel, whenever the usageState attribute changes.	3.1.3.3.1.4.1	3
OE0386		OE_TC_080	The adminState attribute shall contain the device's admin state value.	3.1.3.3.1.4.2	3
OE0387		OE_TC_080	The adminState attribute shall only allow the setting of LOCKED and UNLOCKED values, where setting "LOCKED" is only effective when the adminState attribute value is UNLOCKED, and setting "UNLOCKED" is only effective when the adminState attribute value is LOCKED or SHUTTING_DOWN.	3.1.3.3.1.4.2	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0388		OE_TC_080	The adminState attribute, upon being commanded to be LOCKED, shall transition from the UNLOCKED to the SHUTTING_DOWN state and set the adminState to LOCKED for its entire aggregation of devices (if it has any).	3.1.3.3.1.4.2	3
OE0389		OE_TC_080	The adminState shall then transition to the LOCKED state when the device's usageState is IDLE and its entire aggregation of devices are LOCKED.	3.1.3.3.1.4.2	3
OE0390	C087, C088, C089, C090, C091	OE_TC_088	The device shall send a StateChangeEventType event to the Incoming Domain Management event channel, whenever the adminState attribute changes.	3.1.3.3.1.4.2	3
OE0395		OE_TC_218	The readonly operationalState attribute shall contain the device's operational state (ENABLED or DISABLED).	3.1.3.3.1.4.3	3
OE0400	C092, C093, C094, C095, C096	OE_TC_058	The device shall send a StateChangeEventType event to the Incoming Domain Management event channel, whenever the operationalState attribute changes.	3.1.3.3.1.4.3	3
OE0401		OE_TC_218	The readonly softwareProfile attribute shall contain a profile element (Profile Descriptor) with a file reference to the SPD file.	3.1.3.3.1.4.4	3
OE0403		OE_TC_218	The readonly label attribute shall contain the device's label.	3.1.3.3.1.4.5	3
OE0404		OE_TC_218	The readonly compositeDevice attribute shall contain the object reference of the aggregate device when this device is a parent device.	3.1.3.3.1.4.6	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0405		OE_TC_081	The allocateCapacity operation shall reduce the current capacities of the device based upon the input capacities parameter, when the device's adminState is UNLOCKED, device's operationalState is ENABLED, and device's usageState is not BUSY.	3.1.3.3.1.5.1.3.	3
OE0406		OE_TC_082	The allocateCapacity operation shall set the Device's usageState attribute to BUSY, when the device determines that it is not possible to allocate any further capacity.	3.1.3.3.1.5.1.3.	3
OE0407		OE_TC_004	The allocateCapacity operation shall set the usageState attribute to ACTIVE, when capacity is being used and any capacity is still available for allocation (reference Figure 3-22).	3.1.3.3.1.5.1.3.	3
OE0408		OE_TC_083	The allocateCapacity operation shall return TRUE, if the capacities have been allocated, or FALSE, if not allocated.	3.1.3.3.1.5.1.4	3
OE0409		OE_TC_229	The allocateCapacity operation shall raise the InvalidCapacity exception, when the input capacities parameter contains invalid properties or when attributes of those CF Properties contain an unknown id or a value of the wrong data type.	3.1.3.3.1.5.1.5	3
OE0410		OE_TC_229	The allocateCapacity operation shall raise the InvalidState exception, when the Device's adminState is not UNLOCKED or operationalState is DISABLED.	3.1.3.3.1.5.1.5	3
OE0411		OE_TC_084	The deallocateCapacity operation shall adjust the current capacities of the device based upon the input capacities parameter.	3.1.3.3.1.5.2.3	3
OE0412		OE_TC_085	The deallocateCapacity operation shall set the usageState attribute to ACTIVE when, after adjusting capacities, any of the device's capacities are still being used.	3.1.3.3.1.5.2.3	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0413		OE_TC_085	The deallocateCapacity operation shall set the usageState attribute to IDLE when, after adjusting capacities, none of the device's capacities are still being used.	3.1.3.3.1.5.2.3	3
OE0414		OE_TC_085	The deallocateCapacity operation shall set the adminState attribute to LOCKED as specified in 3.1.3.2.4.4.2.	3.1.3.3.1.5.2.3	3
OE0415		OE_TC_086	The deallocateCapacity operation shall raise the InvalidCapacity exception, when the capacity ID is invalid or the capacity value is the wrong type.	3.1.3.3.1.5.2.5	3
OE0417		OE_TC_230	The deallocateCapacity operation shall raise the InvalidState exception, when the device's adminState is LOCKED or operationalState is DISABLED.	3.1.3.3.1.5.2.5	3
OE0420		OE_TC_087	The releaseObject operation shall cause the device to be unavailable and released from the CORBA environment when the Device adminState attribute transitions to LOCKED.	3.1.3.3.1.5.3.3	3
OE0423		OE_TC_089	The releaseObject operation shall raise the ReleaseError exception when releaseObject is not successful in releasing a logical device due to internal processing errors that occurred within the device being released.	3.1.3.3.1.5.3.5	3
OE0424		OE_TC_130	The error number shall indicate a CF ErrorNumberType.	3.1.3.3.2.3.3	3
OE0431		OE_TC_133	The load operation shall raise the CF InvalidFileName exception when the file designated by the input filename parameter cannot be found.	3.1.3.3.2.5.1.5	3
OE0432		OE_TC_130	The load operation shall raise the LoadFail exception when an attempt to load the device is unsuccessful.	3.1.3.3.2.5.1.5	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0436		OE_TC_134	The unload operation shall raise the CF InvalidFileName exception when the file designated by the input filename parameter cannot be found.	3.1.3.3.2.5.2.5	3
OE0438		OE_TC_145	The errorNumber parameter shall indicate a CF ErrorNumberType value.	3.1.3.3.3.3.1	3
OE0442		OE_TC_221	The value for a stack size shall be an unsigned long.	3.1.3.3.3.3.6	3
OE0443		OE_TC_221	The value for a priority shall be an unsigned long.	3.1.3.3.3.3.7	3
OE0444		OE_TC_222	The error number shall indicate a CF ErrorNumberType value.	3.1.3.3.3.3.8	3
OE0445		OE_TC_227	The execute operation shall execute the function or file identified by the input name parameter using the input parameters and options parameters.	3.1.3.3.3.5.1.3	3
OE0446		OE_TC_227	The execute operation shall convert the input parameters (id/value string pairs) parameter to the standard argv of the POSIX exec family of functions, where argv(0) is the function name.	3.1.3.3.3.5.1.3	3
OE0447		OE_TC_227	The execute operation shall map the input parameters parameter to argv starting at index 1 as follows, argv (1) maps to input parameters (0) id and argv (2) maps to input parameters (0) value and so forth.	3.1.3.3.3.5.1.3	3
OE0448		OE_TC_227	The execute operation shall use these options, when specified, to set the operating system's process/thread stack size and priority, for the executable image of the given input name parameter.	3.1.3.3.3.5.1.3	3
OE0450		OE_TC_222	The execute operation shall raise the InvalidState exception if upon entry the device's adminState attribute is either LOCKED or SHUTTING_DOWN or its operationalState attribute is DISABLED.	3.1.3.3.3.5.1.5	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0451		OE_TC_222	The execute operation shall raise the InvalidFunction exception when the function indicated by the input name parameter does not exist for the device.	3.1.3.3.3.5.1.5	3
OE0452		OE_TC_135	The execute operation shall raise the CF InvalidFileName exception when the file name indicated by the input name parameter does not exist for the device.	3.1.3.3.3.5.1.5	3
OE0453		OE_TC_222	The execute operation shall raise the InvalidParameters exception when the input parameter ID or value attributes are not valid strings.	3.1.3.3.3.5.1.5	3
OE0454		OE_TC_222	The execute operation shall raise the InvalidOptions exception when the input options parameter does not comply with sections 3.1.3.3.3.3.6 STACK_SIZE_ID and 3.1.3.3.3.3.7 PRIORITY_ID.	3.1.3.3.3.5.1.5	3
OE0455		OE_TC_222	The execute operation shall raise the ExecuteFail exception when the operating system “execute” function for the device is not successful.	3.1.3.3.3.5.1.5	3
OE0458		OE_TC_145	The terminate operation shall raise the InvalidProcess exception when the process Id does not exist for the device.	3.1.3.3.3.5.2.5	3
OE0459		OE_TC_022	The readonly devices attribute shall contain a list of devices that have been added to this device or a sequence length of zero if the device has no aggregation relationships with other devices.	3.1.3.3.4.4.1	3
OE0460		OE_TC_023	The addDevice operation shall add the input associatedDevice parameter to the AggregateDevice’s devices attribute when the associatedDevice does not exist in the devices attribute.	3.1.3.3.4.5.1.3	3

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0461		OE_TC_025	The addDevice operation shall write a FAILURE_ALARM log record, upon unsuccessful adding of an associatedDevice to the AggregateDevice's devices attribute.	3.1.3.3.4.5.1.3	3
OE0462		OE_TC_027	The addDevice operation shall raise the CF InvalidObjectReference when the input associatedDevice parameter is a nil CORBA object reference.	3.1.3.3.4.5.1.5	3
OE0463		OE_TC_028	The removeDevice operation shall remove the input associatedDevice parameter from the AggregateDevice's devices attribute.	3.1.3.3.4.5.2.3	3
OE0464		OE_TC_029	The removeDevice operation shall write a FAILURE_ALARM log record, upon unsuccessful removal of the associatedDevice from the AggregateDevice devices attribute.	3.1.3.3.4.5.2.3	3
OE0465		OE_TC_030	The removeDevice operation shall raise the CF InvalidObjectReference when the input associatedDevice parameter is a nil CORBA object reference or does not exist in the AggregateDevice devices attribute.	3.1.3.3.4.5.2.5	3
OE0466		OE_TC_034	The readonly identifier attribute shall contain the instance-unique identifier for a device manager.	3.1.3.2.4.4.1	2
OE0467		OE_TC_034	The identifier shall be identical to the deviceconfiguration element id attribute of the device manager's Device Configuration Descriptor (DCD) file.	3.1.3.2.4.4.1	2
OE0468		OE_TC_224	The readonly label attribute shall contain the device manager's label.	3.1.3.2.4.4.2	2
OE0469		OE_TC_224	The readonly fileSys attribute shall contain the FileSystem associated with this device manager.	3.1.3.2.4.4.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0470		OE_TC_224	The readonly deviceConfigurationProfile attribute shall contain a profile element (Profile Descriptor) with a file reference to the device manager's Device Configuration Descriptor (DCD) file.	3.1.3.2.4.4.4	2
OE0471		OE_TC_224	The readonly registeredDevices attribute shall contain a list of devices that have registered with this device manager or a sequence length of zero if no devices have registered with the device manager.	3.1.3.2.4.4.5	2
OE0472		OE_TC_224	The readonly registeredServices attribute shall contain a list of services that have registered with this device manager or a sequence length of zero if no services have registered with the device manager.	3.1.3.2.4.4.6	2
OE0473		OE_TC_033	The device manager upon start up shall register itself with a domain manager.	3.1.3.2.4.5	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0474		OE_TC_034	A device manager shall use the information in the device manager's DCD for determining: 1. Services to be deployed for this device manager (for example, log(s)), 2. Devices to be created for this device manager (when the DCD deployondevice element is not specified then the DCD componentinstantiation element is deployed on the same hardware device as the device manager), 3. Devices to be deployed on (executing on) another device, 4. Devices to be aggregated to another device, 5. Mount point names for file systems, 6. The DeviceManager's identifier attribute value which is the DCD's id attribute value, and 7. The DeviceManager's label attribute value, which is the DCD's name attribute value.	3.1.3.2.4.5	2
OE0475		OE_TC_283	The device manager shall create FileSystem components implementing the FileSystem interface for each OS file system.	3.1.3.2.4.5	4
OE0476	C109	OE_TC_283	If multiple file systems are to be created, the device manager shall mount created file systems to a FileManager component (widened to a FileSystem through the FileSys attribute).	3.1.3.2.4.5	4
OE0481		OE_TC_285	The device manager shall use the componentinstantiation element's SPD implementation code's stacksize and priority elements, when specified, for the execute operation options parameters.	3.1.3.2.4.5	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0482		OE_TC_285	The device manager shall initialize and then configure logical devices that are started by the device manager, after they have successfully registered with the device manager.	3.1.3.2.4.5	2
OE0483		OE_TC_285	The device manager shall configure a DCD's componentinstantiation element provided the componentinstantiation element has "configure" readwrite or writeonly properties with values.	3.1.3.2.4.5	2
OE0504		OE_TC_090	The getComponentImplementationId operation shall return the SPD implementation element's id attribute that matches the SPD implementation element used to create the component identified by the input componentInstantiationId parameter.	3.1.3.2.4.6.6.4	2
OE0505		OE_TC_090	The getComponentImplementationId operation shall return an empty string when the input componentInstantiationId parameter does not match the id attribute of any SPD implementation element used to create the component.	3.1.3.2.4.6.6.4	2
OE0506		OE_TC_091	Framework Services Interfaces shall be implemented using the CF IDL presented in Appendix C.	3.1.3.4	4
OE0507		OE_TC_066, OE_TC_146	The error number shall indicate a CF ErrorNumberType value.	3.1.3.4.1.3.1	4
OE0509		OE_TC_265	The readonly fileName attribute shall contain the pathname used as the input fileName parameter of the FileSystem::create operation when the file was created.	3.1.3.4.1.4.1	4
OE0510		OE_TC_265	The readonly filePointer attribute shall contain the current file position.	3.1.3.4.1.4.2	4
OE0515		OE_TC_066	The read operation shall raise the IOException when a read error occurs.	3.1.3.4.1.5.1.5	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0519	C111	OE_TC_146	The write operation shall raise the IOException when a write error occurs.	3.1.3.4.1.5.2.5	4
OE0521		OE_TC_147	The sizeOf operation shall raise the CF FileException when a file-related error occurs (e.g., file does not exist anymore).	3.1.3.4.1.5.3.5	4
OE0522		OE_TC_067	The close operation shall release any OE file resources associated with the component.	3.1.3.4.1.5.4.3	4
OE0523		OE_TC_067	The close operation shall make the file unavailable to the component.	3.1.3.4.1.5.4.3	4
OE0524		OE_TC_068	The close operation shall raise the CF FileException when it cannot successfully close the file.	3.1.3.4.1.5.4.5	4
OE0526		OE_TC_069	The setFilePointer operation shall raise the CF FileException when the file pointer for the referenced file cannot be set to the value of the input filePointer parameter.	3.1.3.4.1.5.5.5	4
OE0528		OE_TC_031	At a minimum, the file system shall support name, kind, and size information for a file.	3.1.3.4.2.3.3	4
OE0529		OE_TC_109	For this property, the identifier is CREATED_TIME_ID and the value shall be an unsigned long long data type containing the number of seconds since 00:00:00 UTC, Jan. 1, 1970.	3.1.3.4.2.3.6	4
OE0530		OE_TC_110	For this property, the identifier is MODIFIED_TIME_ID and the value shall be an unsigned long long data type containing the number of seconds since 00:00:00 UTC, Jan. 1, 1970.	3.1.3.4.2.3.7	4
OE0531		OE_TC_111	For this property, the identifier is LAST_ACCESS_TIME_ID and the value shall be an unsigned long long data type containing the number of seconds since 00:00:00 UTC, Jan. 1, 1970.	3.1.3.4.2.3.8	4
OE0533		OE_TC_136	The remove operation shall raise the CF InvalidFileName exception when the input fileName parameter is not a valid absolute pathname.	3.1.3.4.2.5.1.5	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0534		OE_TC_105	The remove operation shall raise the CF FileException when a file-related error occurs.	3.1.3.4.2.5.1.5	4
OE0535		OE_TC_106	The copy operation shall copy the source file identified by the input sourceFileName parameter to the destination file identified by the input destinationFileName parameter.	3.1.3.4.2.5.2.3	4
OE0536		OE_TC_148	The copy operation shall raise the CF FileException exception when a file-related error occurs.	3.1.3.4.2.5.2.5	4
OE0537		OE_TC_137	The copy operation shall raise the CF InvalidFileName exception when the sourceFileName or destinationFileName input parameters are not a valid absolute pathnames.	3.1.3.4.2.5.2.5	4
OE0538		OE_TC_138	The exists operation shall check to see if a file exists based on the fileName parameter.	3.1.3.4.2.5.3.3	4
OE0539		OE_TC_138	The exists operation shall return TRUE if the file exists, or FALSE if it does not.	3.1.3.4.2.5.3.4	4
OE0540		OE_TC_159	The exists operation shall raise the CF InvalidFileName exception when input fileName parameter is not a valid absolute pathname.	3.1.3.4.2.5.3.5	4
OE0542		OE_TC_031	The list operation shall support the "*" and "?" wildcard characters (used to match any sequence of characters (including null) and any single character, respectively.	3.1.3.4.2.5.4.3	4
OE0543		OE_TC_031	The list operation shall return a FileInformationSequence for files that match the search pattern specified in the input pattern parameter.	3.1.3.4.2.5.4.4	4
OE0545		OE_TC_103	The list operation shall raise the CF InvalidFileName exception when the input pattern parameter is not an absolute pathname or cannot be interpreted due to unexpected characters.	3.1.3.4.2.5.4.5	4
OE0546		OE_TC_104	The list operation shall raise the CF FileException when a file-related error occurs.	3.1.3.4.2.5.4.5	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0547		OE_TC_107	The create operation shall create a new File based upon the input fileName parameter.	3.1.3.4.2.5.5.3	4
OE0550		OE_TC_149	The create operation shall raise the CF FileException if the file already exists or another file error occurred.	3.1.3.4.2.5.5.5	4
OE0551		OE_TC_139	The create operation shall raise the CF InvalidFileName exception when the input fileName parameter is not a valid absolute pathname.	3.1.3.4.2.5.5.5	4
OE0552		OE_TC_273	The open operation shall open the file referenced by the input fileName parameter.	3.1.3.4.2.5.6.3	4
OE0555		OE_TC_273	The open operation shall open the file for write access when the input read_Only parameter is FALSE.	3.1.3.4.2.5.6.3	4
OE0556		OE_TC_273	The open operation shall return a File instance on successful completion.	3.1.3.4.2.5.6.4	4
OE0558		OE_TC_150	The open operation shall raise the CF FileException if the file does not exist or another file error occurred.	3.1.3.4.2.5.6.5	4
OE0559		OE_TC_140	The open operation shall raise the CF InvalidFileName exception when the input fileName parameter is not a valid absolute pathname.	3.1.3.4.2.5.6.5	4
OE0560		OE_TC_114	The mkdir operation shall create a file system directory based on the directoryName given.	3.1.3.4.2.5.7.3	4
OE0561		OE_TC_114	The mkdir operation shall create all parent directories required to create the directoryName path given.	3.1.3.4.2.5.7.3	4
OE0562		OE_TC_151	The mkdir operation shall raise the CF FileException if the directory indicated by the input directoryName parameter already exists or if a file-related error occurred during the operation.	3.1.3.4.2.5.7.5	4
OE0563		OE_TC_141	The mkdir operation shall raise the CF InvalidFileName exception when the directoryName is not a valid directory name.	3.1.3.4.2.5.7.5	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0564		OE_TC_115	The rmdir operation shall remove the directory identified by the input directoryName parameter.	3.1.3.4.2.5.8.3	4
OE0565		OE_TC_152	The rmdir operation shall raise the CF FileException when the directory identified by the input directoryName parameter does not exist, the directory contains files, or an error occurs which prohibits the directory from being deleted.	3.1.3.4.2.5.8.5	4
OE0566		OE_TC_142	The rmdir operation shall raise the CF InvalidFileName exception when the input directoryName parameter is not a valid path prefix.	3.1.3.4.2.5.8.5	4
OE0567		OE_TC_275	The query operation shall return file system information to the calling client based upon the given fileSystemProperties' ID.	3.1.3.4.2.5.9.3	4
OE0568		OE_TC_275	The FileSystem::query operation shall recognize and provide the designated return values for the following fileSystemProperties (section 3.1.3.4.2.3.2): 1. SIZE - an ID value of "SIZE" causes the query operation to return an unsigned long long containing the file system size (in octets). 2. AVAILABLE SPACE - an ID value of "AVAILABLE SPACE" causes the query operation to return an unsigned long long containing the available space on the file system (in octets).	3.1.3.4.2.5.9.3	4
OE0569		OE_TC_275	The query operation shall raise the UnknownFileSystemProperties exception when the given file system property is not recognized.	3.1.3.4.2.5.9.5	4
OE0573		OE_TC_276	The mount operation shall associate the specified file system with the mount point referenced by the input mountPoint parameter.	3.1.3.4.3.5.1.3	4
OE0574		OE_TC_276	A mount point name shall begin with a "/" (forward slash character).	3.1.3.4.3.5.1.3	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0575		OE_TC_143	The mount operation shall raise the CF InvalidFileName exception when the input mount point does not conform to the file name syntax in section 3.1.3.4.2.1.	3.1.3.4.3.5.1.5	4
OE0576		OE_TC_276	The mount operation shall raise the MountPointAlreadyExists exception when the mount point already exists in the file manager.	3.1.3.4.3.5.1.5	4
OE0577		OE_TC_276	The mount operation shall raise the InvalidFileSystem exception when the input FileSystem is a null object reference.	3.1.3.4.3.5.1.5	4
OE0582		OE_TC_031	A file manager shall implement the inherited FileSystem operations as required under section 3.1.3.4.2 for each mounted file system.	3.1.3.4.3.5.4	4
OE0583		OE_TC_031	The FileSystem operations inherited by a file manager shall remove the name of the mounted file system from input pathnames before passing the pathnames to any operation on a mounted file system.	3.1.3.4.3.5.4	4
OE0588		OE_TC_070	Domain Profile files shall be complaint to the Document Type Definitions (DTDs) provided in Appendix D [Reference 4].	3.1.3.5	5
OE0589		OE_TC_117	DTD files are installed in the domain and shall have ".dtd" as their filename extension.	3.1.3.5	5
OE0590		OE_TC_070	All XML files shall have as the first two lines as an XML declaration (?xml) and a document type declaration (!DOCTYPE).	3.1.3.5	5
OE0591		OE_TC_070	A Software Package Descriptor file shall have a ".spd.xml" extension.	3.1.3.5.1	5
OE0592		OE_TC_070	A Software Component Descriptor file shall have a ".scd.xml" extension.	3.1.3.5.2	5
OE0594		OE_TC_070	A Properties File shall have a ".prf.xml" extension.	3.1.3.5.4	5
OE0595		OE_TC_070	A Device Package Descriptor File shall have a ".dpd.xml" extension.	3.1.3.5.5	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0596		OE_TC_070	A Device Configuration Descriptor file shall have a ".dcd.xml" extension.	3.1.3.5.6	5
OE0597		OE_TC_070	A DomainManager Configuration Descriptor file shall have a ".dmd.xml" extension.	3.1.3.5.8	5
OE0598		OE_TC_068, OE_TC_069, OE_TC_104, OE_TC_105, OE_TC_147, OE_TC_148, OE_TC_149, OE_TC_150, OE_TC_151, OE_TC_152	The error number shall indicate a CF ErrorNumberType value.	3.1.3.6.3	4
OE0599		OE_TC_103, OE_TC_132, OE_TC_133, OE_TC_134, OE_TC_135, OE_TC_136, OE_TC_137, OE_TC_139, OE_TC_140, OE_TC_141, OE_TC_142, OE_TC_143, OE_TC_144, OE_TC_159	The CF InvalidFileName exception indicates an invalid file name was passed to a file service operation. The error number shall indicate a CF ErrorNumberType value.	3.1.3.6.4	4,2,3
OE0605	C123, C124	OE_TC_060	Valid individual filenames and directory names shall be 40 characters or less.	3.1.3.4.2.1	4
OE0613	AP: C145, C148-C176; OE: C148-C166; C178-C181	OE_TC_119	An application, each application component, and each device manager shall be accompanied by the appropriate Domain Profile files per section 3.1.3.5.	3.2.1.3	5
OE0614		OE_TC_075	Interfaces provided by a component shall be described in a Software Component Descriptor file as provides ports.	3.2.2	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0615		OE_TC_076	Interfaces required by a component shall be described in a Software Component Descriptor file as uses ports.	3.2.2	5
OE0618		OE_TC_092	The executable parameters of a logical device shall accept the standard argv arguments as used in the POSIX exec family of functions.	3.3.1	3
OE0619		OE_TC_092	A logical device shall accept the executable parameters as specified in section 3.1.3.3.3.5.1.3 (ExecutableDevice::execute).	3.3.1	3
OE0620		OE_TC_094	Logical devices shall be limited to using CORBA and CORBA services defined in the referenced minimumCORBA specification [Reference 5].	3.3.2	3
OE0621		OE_TC_095	A logical device shall register itself with a device manager using the value associated with the DEVICE_MGR_IOR parameter per 3.1.3.2.4.5.	3.3.3	3
OE0622		OE_TC_096	A child device shall add itself to a parent device using the executable Composite Device IOR parameter per 3.1.3.2.4.5.	3.3.3	3
OE0623		OE_TC_097	The values associated with the parameters (PROFILE_NAME, COMPOSITE_DEVICE_IOR, DEVICE_ID and DEVICE_LABEL) as described in 3.1.3.2.4.5 shall be used to set the Device's softwareProfile, compositeDevice, identifier, and label attributes, respectively.	3.3.3	3
OE0627		OE_TC_098	For each logical device, allocation properties shall be defined in its referenced SPD's property file.	3.3.4	3
OE0628		OE_TC_099	Software developed for an SCA-compliant system shall be developed in a standard higher order language.	3.4	5
OE0630		OE_TC_100	Legacy software shall interface with the Core Framework in accordance with this specification.	3.4	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0633		OE_TC_101	Hardware critical interfaces shall be defined in Interface Control Documents that are available to other parties without restriction.	3.3.3	5
OE0657		OE_TC_131	The functions in Table B-3 shall behave as described in the applicable clauses of the referenced POSIX referenced POSIX specifications contained in Table B-1.	B.4.1.1	5
OE0658		OE_TC_131	The functions listed in Table B-4 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.2	5
OE0659		OE_TC_131	The functions listed in Table B-5 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.3	5
OE0660		OE_TC_131	The functions listed in Table B-6 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.4	5
OE0661		OE_TC_021	An application that conforms to the AEP shall not result in abnormal termination of the process because this profile does not support multiple processes.	B.4.1.4	5
OE0663		OE_TC_131	The functions listed in Table B-8 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.6	5
OE0664		OE_TC_131	The functions listed in Table B-9 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.7	5
OE0665		OE_TC_131	The functions listed in Table B-10 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.8	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0666		OE_TC_044	An application that conforms to the AEP shall be guaranteed that the file mode creation mask for any object created by any process is S-IRWXU; that is, the object shall be fully accessible to the creator.	B.4.1.8	5
OE0667		OE_TC_131	The functions listed in Table B-11 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1	B.4.1.9	5
OE0668		OE_TC_131	The functions listed in Table B-12 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.10	5
OE0669		OE_TC_131	The functions listed in Table B-13 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.11	5
OE0670		OE_TC_131	The functions listed in Table B-14 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.12	5
OE0671		OE_TC_131	The function listed in Table B-15 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.13	5
OE0672		OE_TC_131	The function listed in Table B-16 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.14	5
OE0673		OE_TC_131	The functions listed in Table B-17 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.15	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0700	C182 - C240	OE_TC_005, OE_TC_006, OE_TC_007, OE_TC_008, OE_TC_009, OE_TC_010, OE_TC_011, OE_TC_012, OE_TC_013, OE_TC_014, OE_TC_015, OE_TC_016, OE_TC_017, OE_TC_018, OE_TC_019	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification [Reference 7].	3.1.2.2	1
OE0701		OE_TC_062	Base Device Interfaces shall be implemented using the CF IDL presented in Appendix C.	3.1.3.3	3
OE0702		OE_TC_003	The OE shall provide two standard event channels: Incoming Domain Management and Outgoing Domain Management.	3.1.2.3.1	5
OE0703		OE_TC_102	Base Application Interfaces shall be implemented using the CF IDL presented in Appendix C.	3.1.3.1	1
OE0707		OE_TC_121	The create operation shall perform the comparison of allocation properties of the application to those of each candidate device, according to the allocation property's action element, for those application component properties whose kindtype is allocation and whose action element is not external.	3.1.3.2.2.5.1.3	2
OE0708		OE_TC_162	The create operation shall deallocate any capacity allocations on devices that do not satisfy the application components allocation requirements or that are not utilized due to an unsuccessful application creation.	3.1.3.2.2.5.1.3	2
OE0710		OE_TC_164	The create operation shall establish connections for an application which are specified in the SAD domainfinder element.	3.1.3.2.2.5.1.3	2

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0715		OE_TC_166	The registerDevice operation shall verify that the input parameters, registeringDevice and registeredDeviceMgr, are not nil CORBA object references.	3.1.3.2.3.6.2.3	2
OE0716		OE_TC_167	The registerDevice operation shall return without exception and not register a new device when that device, indicated by the input registeringDevice parameter, has the same identifier attribute as a previously registered device and the reference to the registered device refers to an existing object.	3.1.3.2.3.6.2.3	2
OE0717		OE_TC_168	The registerDevice operation shall register the new device indicated by the input registeringDevice parameter, when the previously registered device has the same identifier attribute as the new device and the reference to the registered device refers to a nonexistent object.	3.1.3.2.3.6.2.3	2
OE0718		OE_TC_168	The registerDevice operation shall write an ADMINISTRATIVE_EVENT log record when reference to the registered device refers to a nonexistent object.	3.1.3.2.3.6.2.3	2
OE0719		OE_TC_113	The registerDevice operation shall establish any pending connections from previously registered device managers when the registering device completes these connections.	3.1.3.2.3.6.2.3	2
OE0720		OE_TC_156	The domain manager shall create a naming context using "/DomainName" as the id attribute to the input name parameter, and "" (Null string) as the kind attribute.	3.1.3.2.3.5	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0722		OE_TC_026	The installApplication operation shall raise the ApplicationAlreadyInstalled exception when the softwareassembly element id attribute of the referenced application is the same as a previously registered application.	3.1.3.2.3.6.3.5	2
OE0725		OE_TC_073	The registerService operation shall register the new service, indicated by the input registeringService parameter, when the previously registered service has the same name and type as the new service and the reference to the registered service refers to a nonexistent object.	3.1.3.2.3.6.7.3	2
OE0729		OE_TC_218	The readonly compositeDevice attribute shall contain a nil CORBA object reference when this device is not a parent device.	3.1.3.3.1.4.6	3
OE0730		OE_TC_118	The allocateCapacity operation shall only accept properties for the input capacities parameter which are simple properties whose kindtype is allocation and whose action element is external contained in the component's SPD.	3.1.3.3.1.5.1.3	3
OE0731		OE_TC_153	Multiple loads of the same file as indicated by the input fileName parameter shall not result in an exception.	3.1.3.3.2.5.1.3	3
OE0732		OE_TC_024	The registerDeviceManager operation shall raise the CF InvalidProfile exception when the device manager's DCD file and the DCD's referenced files do not exist.	3.1.3.2.3.6.1.5	2
OE0733	C125	OE_TC_060	A valid pathname shall not exceed 1024 characters.	3.1.3.4.2.1	4
OE0734		OE_TC_106	The copy operation shall overwrite the destination file, when the destination file already exists and is not identical to the source file.	3.1.3.4.2.5.2.3	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0735		OE_TC_144	The copy operation shall raise the CF InvalidFileName exception when the destination pathname is identical to the source pathname.	3.1.3.4.2.5.2.5	4
OE0736		OE_TC_031	These wildcards shall only be applied following the right-most forward-slash character ("/") in the pathname contained in the input pattern parameter.	3.1.3.4.2.5.4.3	4
OE0737		OE_TC_273	The open operation shall set the filePointer attribute of the returned file instance to the beginning of the file.	3.1.3.4.2.5.6.4	4
OE0739		OE_TC_040	The file manager shall propagate exceptions raised by a mounted file system.	3.1.3.4.3.5.4	4
OE0741		OE_TC_077	All non-standard interfaces shall be defined in Interface Control Documents that are available to other parties without restriction to the extent that interfacing or replacement hardware and software can be developed by other parties without restriction.	3.2.2	5
OE0742		OE_TC_077	All SCA APIs shall have their interfaces described in IDL.	3.2.2.1	5
OE0743		OE_TC_077	All non-IDL interfaces shall provide an IDL mapping within the service definition.	3.2.2.1	5
OE0745		OE_TC_156	The domain manager shall create a name binding to the created naming context using "/DomainName" as the id attribute to the input name parameter, and "" (Null string) as the kind attribute, where DomainName is identical to the name attribute of the domain manager's DMD domainmanagerconfiguration element and the input object parameter is the domain manager object reference. [Reference 6]	3.1.3.2.3.5	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0746		OE_TC_001	The OE shall provide an implementation of a CO bind_new_context, unbind, destroy, and resolve as defined in the OMG Interoperable Naming Service Specification [Reference 6] using the IDL found in Appendix A of that reference.	3.1.2.1	5
OE0747		OE_TC_002	Log producers shall implement a configure property which is a CF Properties type with an id of "PRODUCER_LOG_LEVEL" and a value that is a CosLwLog::LogLevelSequence.	3.1.2.2.1	5
OE0750		OE_TC_155	The execute parameter for the Naming Context IOR shall be a CF Properties type with an id element set to "NAMING_CONTEXT_IOR" and a value element set to the stringified IOR of the naming context to which the component will bind.	3.1.3.2.2.5.1.3	5
OE0751		OE_TC_160	The Name Binding execute parameter shall be a CF Properties type with an id element set to "NAME_BINDING" and a value element set to a string in the format of "ComponentName_UniqueIdentifier".	3.1.3.2.2.5.1.3	5
OE0752		OE_TC_161	The Component Identifier execute parameter shall be a CF Properties type with an id element set to "COMPONENT_IDENTIFIER" and a value element set to a string in the format of "Component_Instantiation_Identifier: Application_Name".	3.1.3.2.2.5.1.3	5
OE0754		OE_TC_112	The device manager shall pass "execparam" parameters' IDs and values as string values.	3.1.3.2.4.5	2
OE0755		OE_TC_031	The list operation shall return a zero length sequence when no file is found which matches the search pattern.	3.1.3.4.2.5.4.4	4

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0756		OE_TC_112	The device manager shall pass the componentinstantiation element "execparam" properties that have values as parameters.	3.1.3.2.4.5	2
OE0760		OE_TC_283	Each mounted file system name shall be unique within the device manager.	3.1.3.2.4.5	4
OE0761		OE_TC_273	The open operation shall open the file with read-only access when the input read_Only parameter is TRUE.	3.1.3.4.2.5.6.3	4
OE0762		OE_TC_131	The functions listed in Table B-7 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.5	5
OE0763		OE_TC_131	The functions listed in Table B-18 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.16	5
OE0764		OE_TC_131	The functions listed in Table B-19 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.17	5
OE0765		OE_TC_131	The functions listed in Table B-20 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.18	5
OE0767		OE_TC_131	The functions listed in Table B-21 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.19	5
OE0768		OE_TC_131	The functions listed in Table B-22 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.20	5
OE0769		OE_TC_131	The options, limits, and any other constraints on POSIX.1 shall be provided as described in Table B-2.	B.4.1	5

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0770		OE_TC_131	The function listed in Table B-23 shall behave as described in the referenced clause.	B.4.1.21	5
OE0771		OE_TC_176	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.	2.1 Extension	App D
OE0772		OE_TC_176	The registerService operation shall, upon successful service registration of a non-SCA service with an input name parameter in the "identifier\type" format, make the value provided in the "type" potion of the name accessible via the domainfinder servicetype mechanism.	2.1 Extension	App D
OE0773		OE_TC_177	The unregisterService operation shall remove non-SCA services (i.e. those with a name in the "identifier\type" format) by matching either a fully qualified name in the "identifier\type" format or a simple name with only the "identifier" portion.	2.2 Extension	App D
OE0774		OE_TC_178	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.	2.3 Extension	App D
OE0775		OE_TC_178	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	2.3 Extension	App D

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
			Deployment Platform Descriptor file.		
OE0776		OE_TC_178	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.	2.3 Extension	App D
OE0777		OE_TC_178	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.	2.3 Extension	App D
OE0778		OE_TC_178	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.	2.3 Extension	App D
OE0779		OE_TC_179	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	2.3 Extension	App D

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
			implementation provides enhanced deployment support via the use of an Application Deployment Descriptor file.		
OE0780		OE_TC_180	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.	2.3 Extension	App D
OE0781		OE_TC_181	The create operation shall raise the InvalidInitConfiguration exception when the input initConfiguration parameter "DEPLOYMENT_CHANNEL" property contains an invalid channel reference.	2.4 Extension	App D
OE0782		OE_TC_181	The InvalidInitConfiguration invalidProperties parameter shall identify the invalid channels.	2.4 Extension	App D
OE0783		OE_TC_182	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.	2.4 Extension	App D

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0784		OE_TC_183	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 cannot be established to a service identified within that list.	2.4 Extension	App D
OE0785		OE_TC_184	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.	2.5 Extension	App D
OE0786		OE_TC_184	The device manager shall pass the componentinstantiation element "execparam" properties that have values as parameters.	2.5 Extension	App D
OE0787		OE_TC_184	The device manager shall pass "execparam" parameters' IDs and values as string values.	2.5 Extension	App D
OE0788		OE_TC_185	A Deployment Platform Descriptor file shall have a ".pdd.xml" extension.	2.8 Extension	App D

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0790		OE_TC_172	The function listed in Table B-24 shall behave as described in the referenced clause.	B.4.1.22	App C
OE0791		OE_TC_175	The Standard C Library header files listed in Table B-25 shall be included within the AEP as described in the referenced clause.	B.5	App C
OE0794		OE_TC_172	The functions in Table B-3 shall behave as described in the applicable clauses of the referenced POSIX referenced POSIX specifications contained in Table B-1.	B.4.1.1	App C
OE0795		OE_TC_172	The functions listed in Table B-4 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.2	App C
OE0796		OE_TC_172	The functions listed in Table B-5 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.3	App C
OE0797		OE_TC_172	The functions listed in Table B-6 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.4	App C
OE0798		OE_TC_170	An application that conforms to the AEP shall not result in abnormal termination of the process because this profile does not support multiple processes.	B.4.1.4	App C
OE0800		OE_TC_172	The functions listed in Table B-8 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.6	App C
OE0801		OE_TC_172	The functions listed in Table B-9 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.7	App C

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0802		OE_TC_172	The functions listed in Table B-10 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.8	App C
OE0803		OE_TC_173	An application that conforms to the AEP shall be guaranteed that the file mode creation mask for any object created by any process is S-IRWXU; that is, the object shall be fully accessible to the creator.	B.4.1.8	App C
OE0804		OE_TC_172	The functions listed in Table B-11 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.9	App C
OE0805		OE_TC_172	The functions listed in Table B-12 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.10	App C
OE0806		OE_TC_172	The functions listed in Table B-13 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.11	App C
OE0807		OE_TC_172	The functions listed in Table B-14 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.12	App C
OE0808		OE_TC_172	The function listed in Table B-15 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.13	App C
OE0809		OE_TC_172	The function listed in Table B-16 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.14	App C

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0810		OE_TC_172	The functions listed in Table B-17 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.15	App C
OE0811		OE_TC_172	The functions listed in Table B-7 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.5	App C
OE0812		OE_TC_172	The functions listed in Table B-18 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.16	App C
OE0813		OE_TC_172	The functions listed in Table B-19 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.17	App C
OE0814		OE_TC_172	The functions listed in Table B-20 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.18	App C
OE0815		OE_TC_172	The functions listed in Table B-21 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.19	App C
OE0816		OE_TC_172	The functions listed in Table B-22 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.	B.4.1.20	App C
OE0817		OE_TC_172	The options, limits, and any other constraints on POSIX.1 shall be provided as described in Table B-2.	B.4.1	App C
OE0818		OE_TC_172	The function listed in Table B-23 shall behave as described in the referenced clause.	B.4.1.21	App C

Requirement Tag	Criterion Tag	Test Case Number	Requirement Text	Section	App Volume
OE0819		OE_TC_290	The functions listed in Table 1-2 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table 1-1.	Networking AEP, 1.4.1.1	5
OE0820		OE_TC_290	The functions listed in Table 1-3 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table 1-1.	Networking AEP, 1.4.1.2	5

Appendix A: Acronym List

Acronym	Definition
ADD	Application Deployment Descriptor
AEP	Application Environment Profile
AOESTD	Automated Operating Environment Software Test Description
API	Application Program Interface
BIT	Built-in Test
C/C++	The high level programming languages C and C++
CD	Compact Disc
CF	Core Framework
CORBA	Common Object Request Broker Architecture
DCD	Device Configuration Descriptor
DCE	Distributed Computer Environment
DISA	Defense Information Systems Agency
DMD	DomainManager Configuration Descriptor
DPD	Device Package Descriptor
DCE	Distributed Computer Environment
DISA	Defense Information Systems Agency
DMD	DomainManager Configuration Descriptor
DPD	Device Package Descriptor
DSP	Digital Signal Processor
DTD	Document Type Definition (XML term)
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
ICD	Interface Control Document
ICWG	Interface Control Working Group
ID	Identifier
IDE	Integrated Development Environment

Acronym	Definition
IDL	Interface Definition Language
IDM	Incoming Domain Management
IOR	Interoperable Object Reference
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
JITC	Joint Interoperability Test Command
JTAP	JTNC Test Application
JTEL	JTNC Test and Evaluation Laboratory
JTNC	Joint Tactical Networking Center
JTR	Joint Tactical Radio
LWL	Lightweight Log
MS	Microsoft
N/A	Not Applicable
ODM	Outgoing Domain Management
OE	Operating Environment
PDD	Deployment Platform Descriptor
POSIX	Portable Operating System Interface
PRF	Property Descriptor File
SAD	Software Assembly Descriptor
SCA	Software Communications Architecture
SCD	Software Component Descriptor
SDD	System Design Document
SDR	Software Defined Radio
SPD	Software Package Descriptor
SRD	Support and Rationale Document for the SCA
SRS	Software Requirements Specification
STD	Software Test Description
SVD	Software Version Descriptor
T&E	Test and Evaluation
TC	Test Case
TC2	Technical Corrigendum 2
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
UUT	Unit Under Test

Acronym	Definition
VDD	Version Descriptor Document
XML	eXtensible Markup Language

Appendix B: SCA v2.2.2 OE Requirement Test Procedures

[Appendix B is provided as a separate document.]

Appendix C: AEP Amended Requirements Test Procedures

[Appendix C is provided as a separate document.]

Appendix D: SCA Extensions Requirements Test Procedures

[Appendix D is provided as a separate document.]