Joint Tactical Networking Center Test and Evaluation Laboratory

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

Appendix B Volume 1
Test Procedures for Base Application Interfaces and OMG Lightweight Log

13 April 2022

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

Prepared by:
Joint Tactical Networking Center Test and Evaluation Laboratory

Table of Contents

APPENDIX B VOLUME 1 TEST PROCEDURES B-4

B.1. Volume 1 Base Application Interfaces and OMG Lightweight Log B-6

B.1.1. OE_TC_005 - OMG Lightweight Log Service :: LogStatus B-6

B.1.2. OE_TC_006 - OMG Lightweight Log Service :: get_n_records B-17

B.1.3. OE_TC_007 - OMG Lightweight Log Service :: LogFullAction B-22

B.1.4. OE_TC_008 - OMG Lightweight Log Service :: get_availability_status B-28

B.1.5. OE_TC_009 - OMG Lightweight Log Service :: administrative_state B-34

B.1.6. OE_TC_010 - OMG Lightweight Log Service :: get_operational_state B-42

B.1.7. OE_TC_011 - OMG Lightweight Log Service :: get_record_id_from_time B-48

B.1.8. OE_TC_012 - OMG Lightweight Log Service :: retrieve_records B-53

B.1.9. OE_TC_013 - OMG Lightweight Log Service :: retrieve_records_by_level B-64

B.1.10. OE_TC_014 - OMG Lightweight Log Service :: retrieve_records_by_producer_id B-74

B.1.11. OE_TC_015 - OMG Lightweight Log Service :: retrieve_records_by_producer_name B-87

B.1.12. OE_TC_016 - OMG Lightweight Log Service :: write_records B-99

B.1.13. OE_TC_017 - OMG Lightweight Log Service :: write_record B-113

B.1.14. OE_TC_018 - OMG Lightweight Log Service :: clear_log B-127

B.1.15. OE_TC_019 - OMG Lightweight Log Service :: destroy B-135

B.1.16. OE_TC_020 - Port :: connectPort B-143

B.1.17. OE_TC_032 - Port :: connectPort raises OccupiedPort B-151

B.1.18. OE_TC_035 - Port :: disconnectPort B-157

B.1.19. OE_TC_037 - LifeCycle :: initialize raises InitializeError B-163

B.1.20. OE_TC_038 - LifeCycle :: releaseObject B-168

B.1.21. OE_TC_039 - LifeCycle :: releaseObject raises ReleaseError B-174

B.1.22. OE_TC_041 - TestableObject :: runTest uses testId parameter B-179

B.1.23. OE_TC_042 - TestableObject :: runTest uses testValues parameter B-184

B.1.24. OE_TC_043 - TestableObject :: runTest returns results B-190

B.1.25. OE_TC_045 - TestableObject :: runTest validates testValues parameter B-196

B.1.26. OE_TC_046 - TestableObject :: runTest validates parameters B-201
B.1.27. OE_TC_047 - PropertySet :: configure B-207
B.1.28. OE_TC_048 - PropertySet :: configure property minimums B-213
B.1.29. OE_TC_049 - PropertySet :: configure raises PartialConfiguration B-220
B.1.30. OE_TC_050 - Resource :: start raises StartError B-226
B.1.31. OE_TC_064 - Resource :: stop raises StopError B-231
B.1.32. OE_TC_071 - Resource :: stop B-236
B.1.33. OE_TC_072 - TestableObject :: runTest parameters B-241
B.1.34. OE_TC_093 - TestableObject :: runTest exception parameter B-252
B.1.35. OE_TC_102 - Base Application Interfaces B-258

Index of Test Case Titles for Manual Tests B-265

APPENDIX B VOLUME 1 TEST PROCEDURES

This Appendix contains the test case procedures including the test case name, test objective, preconditions, parameters, test description, the name of the related automated test name (if relevant), and the test requirement(s) being tested.

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

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

The definitions for the various test results are as follows:

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

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

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

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

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

B.1. Volume 1 Base Application Interfaces and OMG Lightweight Log

This volume consists of manual test cases that verify requirements related to SCA Base Application Interfaces and the OMG Lightweight Log.

B.1.1. OE_TC_005 - OMG Lightweight Log Service :: LogStatus

Test Case Number: OE_TC_005

OMG Lightweight Log Service::LogStatus

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Specification
OE0700-C182	This operation(*get_max_size()*) returns the maximum capacity in bytes of the storage area.
OE0700-C183	The *get_current_size()* operation returns the size in bytes of the log storage area currently occupied by logging records.
OE0700-C184	This operation raises the *InvalidParam* exception if the supplied parameter is invalid.
OE0700-C185	This operation(*set_max_size*()) allows setting of the maximum capacity in bytes of the storage area.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 FINAL 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Section 3.3.1.1, page 3-8, Section 3.3.1.2, page 3-9, Section 3.3.4.1 page 3-23.

Test Objective

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. 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.

Places to Verify

Log Service

OMG Log Interfaces

Exceptions

exception InvalidParam { string details; };

Operations

unsigned long get_max_size();
unsigned long long get_current_size();
void set_max_size(in unsigned long long size)
raises (InvalidParam);

Preconditions

- All the Domain profile files are available.
- The source code files are available.

Test Description

A. Determine from the Domain Profile whether log services are provided.(OE0700)
   1. **Pass:** Log service is provided.
   2. **Untested:** No log service is provided.

B. Verify that the *set_max_size()* operation exists and accepts the correct type of parameter(unsigned long long). The operation may, although, exist as a "no-op" function and does not have to be implemented. If implemented, verify that the operation sets the maximum size, in bytes, allowed in logging storage area. (OE0700-C185)
   NOTE: A ceiling value might exist that limits the maximum size of the storage area. The function might not set the logging area's maximum size beyond this ceiling number; this is acceptable.
   1. **Pass:** The *set_max_size()* operation exists. If it is fully implemented, it should allow the maximum size of the logging storage to be set by the *set_max_size()* operation within the constraints of the platform implementation(e.g. a ceiling value).
   2. **Fail:** The *set_max_size()* operation does not exist, or it does not correctly set the maximum size of the logging area.

C. Verify that the *set_max_size()* operation raises the *InvalidParam* exception when the supplied parameter is invalid. If the InvalidParam exception is not thrown, verify that the log is not set greater than a designated ceiling value compatible for the environment. (OE0700-C184)

1. **Pass:** The *set_max_size()* operation raises the *InvalidParam* exception when the parameter supplied is invalid. If the InvalidParam exception is not raised, *the set_max_size()* operation must not set the size of the log greater than a designated ceiling value
2. **Fail:** The *set_max_size()* operation does not raise the *InvalidParam* exception when the parameter supplied is invalid. If the InvalidParam exception is not raised, the requirement fails if the operation is allowed to set the maximum Log to a size greater than the available storage area in the Log, without limiting it a ceiling value.

D. Verify that the *get_max_size()* operation retrieves the maximum size in bytes of the logging storage area. (OE0700-C182)
   1. **Pass:** The *get_max_size()* operation correctly obtains the maximum number, in bytes, allowed in the logging storage area.
   2. **Fail:** The *get_max_size()* operation does not retrieve the correct maximum size, in bytes, allowed in the logging storage area.

E. Verify that the *get_current_size()* operation returns the size, in bytes, of the amount of storage area currently being occupied by the records in the log. (OE0700-C183)
   1. **Pass:** The *get_current_size()* operation correctly obtains the size, in bytes, of the size of the records that currently occupy the logging storage area.
   2. **Fail:** The *get_current_size()* operation does not retrieve the correct size, in bytes, of the records that currently occupy the logging storage area.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination). (OE0700)				
NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). If the OE software engineer is not available, then steps 1 thru 7 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file				
1. Locate and open the DMD file(s) for the OE.	DMD file opens. Note: The DMD file may be named DomainManager.dmd.xml			
2. Determine the log service(s) used, if any, recording the name of the service.	**Pass:** A log service is found. (OE0700) **Untested:** A log service is not found. (OE0700) Note: Most likely, the name of the log service is LogService. In our example, we are using qqqqq.		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> xxxxx </usesidentifier> <findby> <domainfinder type= "log" name="qqqqq"> </domainfinder> </findby> </service> … </services>	
A – cont: Determine from the domain profile whether log services are provided. (OE0700) **Note: This section of Objective A deals specifically with DCD file examination for SPD file, component files, and component placement declarations.**				

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Locate and open the DCD file(s) for the OE.	DCD file opens.		The Domain DCD file may be named DeviceManager.dcd.xml.	
4. Locate log service *componentplacement* declaration in the DCD file using the name of the log service from the DMD and record the *componentfileref refid*.	*componentplacement* declaration for declared log service exists in the DCD file.		The name of the log service will be a usagename in the componentplacement section of the DCD. XML declaration example: <componentplacement> <componentfileref refid="qqqqq_File"> </componentfileref> <componentinstantiation id="xxxxx" <usagename> qqqqq </usagename> </componentinstantiation> </componentplacement>	
5. Locate the log service *componentfile id* declaration *using the componentfileref refid* and record the related SPD file name.	*Componentfile id* declaration for declared log service exists in the DCD file.		The refid from step 4 will be the id in the componentfile section of the DCD. XML declaration example: <componentfiles> <componentfile id=**"qqqqq_File**" type="Software Package Descriptor"> <**localfile name="qqqqq.spd.xml"/>** </componentfile>	

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Examine the SPD files (step 5) and record the localfile name.	log service executable declaration exists.		XML declaration example: <implementation id="xxxx"> <code type= "xxxxx"> <localfile name= "…/qqqqq.exe"/>	
B. Verify that the *set_max_size()* operation exists and accepts the correct type of parameter (unsigned long long). The operation may, although, exist as a "no-op" function and does not have to be implemented. If implemented, verify that the operation sets the maximum size, in bytes, allowed in logging storage area. (OE0700-C185)				
7. Identify the source code for the *set_max_size()* operation.	**Pass:** The *set_max_size()* operation exists. (OE0700-C185) **Fail:** The *set_max_size()* operation does not exist. (OE0700-C185)		Failure of this criterion OE0700-C185 means failure of the requirement OE0700.	
8. Verify that the *set_max_size()* accepts a parameter of type *<unsigned long long>*.	**Pass:** The *set_max_size()* accepts a parameter of type *<unsigned long long>*. (OE0700-C185) **Fail:** The *set_max_size()* does not accept a parameter of type *<unsigned long long>*. (OE0700-C185)		Failure of this criterion OE0700-C185 means failure of the requirement OE0700.	
9. CASE 1: The operation is NO-OP and is empty. Verify that the *set_max_size()* operation is a completely empty function that does not change the maximum size of the log.	**Pass:** The *set_max_size()* is a completely empty function and does not alter the maximum size of the Log in any way. (OE0700-C185) **Fail:** The *set_max_size()* is a completely empty function and alters the maximum size of the Log in any way. (OE0700-C185)		Failure of this criterion OE0700-C185 means failure of the requirement OE0700.	

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. CASE 2: The *set_max_size()* operation is implemented. Verify that the *set_max_size()* operation sets the maximum size (in bytes) allowed in the logging storage area.	The *set_max_size()* operation **Pass:** The *set_max_size()* operation sets the maximum size (in bytes and indicated by the parameter), allowed in the logging storage area. (OE0700-C185) **Fail:** The *set_max_size()* operation does not set the maximum size (in bytes and indicated by the parameter), allowed in the logging storage area. (OE0700-C185)		Failure of this criterion OE0700-C185 means failure of the requirement OE0700.	
C. Verify that the *set_max_size()* operation raises the InvalidParam exception when the supplied parameter is invalid. If the InvalidParam exception is not thrown, verify that the log is not set greater than a designated ceiling value compatible for the environment. (OE0700-C184)				
11. Verify that the *set_max_size()* operation raises the *InvalidParam* exception when the supplied parameter is invalid. If this step cannot verified, proceed to step 12 in the case that operation does not throw an exception but sets the log to a ceiling value.	**Pass:** The *set_max_size()* operation raises the InvalidParam exception when the supplied parameter is invalid. In both the cases stated, the lower bound and the upper bound, the operation can correctly process an invalid parameter. (OE0700-C184) **Fail:** The *set_max_size()* operation does not raise the InvalidParam exception when the supplied parameter is invalid. In both the cases stated, the lower bound and the upper bound, the operation can correctly process an invalid parameter. (OE0700-C184)		Might need to consult developer documentation about the limitations of the parameter. Failure of this criterion OE0700-C184 means failure of the requirement OE0700.	

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
12. If the InvalidParam exception is not thrown, verify that the *set_max_size()* operation sets the size of the Log no greater than a designated ceiling value compatible with the platform specific implementation.	**Pass:** If the InvalidParam exception is not thrown, the *set_max_size()* operation sets the Log greater than a designated ceiling value compatible with the platform specific implementation. (OE0700-C184) **Fail:** If the InvalidParam exception is not thrown, the *set_max_size()* operation does not set the Log greater than a designated ceiling value compatible with the platform specific implementation. (OE0700-C184)		Failure of this criterion OE0700-C184 means failure of the requirement OE0700.	
D. Verify that the *get_max_size()* operation retrieves the maximum size in bytes of the logging storage area. (OE0700-C182)				
13. Identify the source code for *get_max_size()* operation.	**Pass:** The *get_max_size()* operation exists. (OE0700-C182) **Fail:** The *get_max_size()* operation does not exist. (OE0700-C182)		Failure of this criterion OE0700-C182 means failure of the requirement OE0700.	
14. Verify that the *get_max_size()* operation correctly retrieves the maximum size, in bytes, of the logging storage area.	**Pass:** The *get_max_size()* operation retrieves the correct maximum size, in bytes, of the logging storage area. (OE0700-C182) **Fail:** The *get_max_size()* operation does not retrieve the correct maximum size, in bytes, of the logging storage area. (OE0700-C182)		Failure of this criterion OE0700-C182 means failure of the requirement OE0700.	
E. Verify that the *get_current_size()* operation returns the size, in bytes, of the amount of storage area currently being occupied by the records in the log. (OE0700-C183)				

OE_TC_005				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
15. Identify the source code for *get_current_size()* operation.	**Pass:** The *get_current_size()* operation exists. (OE0700-C183) **Fail:** The *get_current_size()* operation does not exist. (OE0700-C183)		Failure of this criterion OE0700-C183 means failure of the requirement OE0700.	
16. Verify that the *get_current_size()* operation returns the correct total size, in bytes, of the current logging records occupying the logging storage area.	**Pass:** The *get_current_size()* operation returns the correct total size, in bytes, of the current logging records occupying the logging storage area. (OE0700-C183) **Fail:** The *get_current_size()* operation does not return the correct total size, in bytes, of the current logging records occupying the logging storage area. (OE0700-C183)		The current size of the log cannot be larger than the maximum size of the storage area retrieved by the get_max_size() operation. Failure of this criterion OE0700-C183 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_005				
DMD file				
DCD file:				
Step 1-5 (DMD/DCD logging services)	Step 6 (Locate source code)	Step 7 (*set_max_size()* operation exists)	Step 8 (*set_max_size()* Parameter)	Step 9 (CASE 1: fully implemented *set_max_size())*
Step 10 (CASE 2: implemented as NO-OP)	*Step 11* (*set_max_size()* InvalidParam exception)	*Step 12* (*set_max_size()*)	*Step 13* (*get_max_size()* exists)	Step 14 (maximum)
Step 15 (*get_current_size()* operation exists)	Step 16 (retrieves actual size of logs)			

Test Summary: OE_TC_005

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s)(x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C182 _____

OE0700-C183 _____

OE0700-C184 _____

OE0700-C185 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.2. OE_TC_006 - OMG Lightweight Log Service :: get_n_records

Test Case Number: OE_TC_006

OMG Lightweight Log Service::get_n_records

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Specification
OE0700-C186	The *get_n_records()* operation returns the number of logging records currently stored in the log storage area.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2, 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005, Version 1.1, formal/05-02-02	Pages 3-9 and 3-10, Section 3.3.1.3

Test Objective

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C186, of the OMG Lightweight Log. 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.

Places to Verify

Log Service

OMG Log Interfaces

Operations

unsigned long long get_n_records();

Preconditions

- All the Domain profile files of the OE are available.
- The source code is available.

Test Description

A. Determine from the domain profile whether log services are provided. (OE0700)
   1. **Pass**: Log service is provided.
   2. **Untested**: No log service is provided.

B. For each log service identified in Step A, verify that the *get_n_records()* operation exists and that it returns the number of records presently stored in the log storage area. (OE0700-C186)
   1. **Pass**: The number of records returned by the *get_n_records()* operation is equal to the current number of records in the Log.
   2. **Fail**: The number of records returned by the *get_n_records()* operation is not equal to the current number of records in the log storage area.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_006				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided. (OE0700)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B. For each log service identified in Step A, verify that the *get_n_records()* operation exists and that it returns the number of records presently stored in the log storage area (OE0700-C186)				
1. Verify that the *get_n_records()* operation exists in the source code.	**Pass:** The *get_n_records()* operation exists. (OE0700-C186) **Fail:** The *get_n_records()* operation does not exist. (OE0700-C186)		Failure of this criterion OE0700-C186 means failure of the requirement OE0700.	
2. Verify that the *get_n_records()* operation will accurately retrieve the number of records currently stored in the logging storage area. NOTE: The number of records could possibly be kept by a static variable or by directly accessing the log storage area. Either way this operation must make sure that both ways accurately report the number of logs in the system.	**Pass:** The *get_n_records()* operation obtains the current number of records in the logging system. (OE-0700-C186) **Fail:** The *get_n_records()* operation does not obtain the current number of records in the logging system. (OE0700-C186)		Failure of this criterion OE0700-C186 means failure of the requirement OE0700.	
End of Test				

Test Recording Log-OE_TC_006			
DMD file			
OE_TC_005, Step 2 (log service)			
DCD file			
OE_TC_005, Step 4-5 (log service)	OE_TC_005, Step 6 (executable)	Step 1 (*get_n_records*() exists)	Step 2 (current number of records)

Test Summary: OE_TC_006

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s)(x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C186 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.3. OE_TC_007 - OMG Lightweight Log Service :: LogFullAction

Test Case Number: OE_TC_007

OMG Lightweight Log Service::LogFullAction

Requirements

SCA v2.2.2 Tag	SCA v2.2.2
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight
OE0700-C187	The *get_log_full_action* operation returns the information about which action the Logging Service will take when the storage area becomes full.
OE0700-C188	The possible values are HALT, which means no further logging records are accepted and stored; or WRAP, which means the Log continues by overwriting the oldest records in the storage area.
OE0700-C189	The *set_log_full_action* operation allows the specification which action should be taken after all free space in the log storage area is depleted.
OE0700-C190	The possible values are HALT, which means no further logging records are accepted and stored; or WRAP, which means the Log continues by overwriting the oldest records in the storage area.
OE0700-C191	When the LogFullAction type is set to WRAP, the Log will set the availability status logFull state to false.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005, Version 1.1, formal/05-02-02	Page 3-11, section 3.3.1.4 Page 3-23, section 3.3.4.2

Test Objective

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.

Places to Verify

Log Service

OMG Log Interfaces

Data

enum LogFullAction {WRAP, HALT};

Operations

LogFullAction get_log_full_action();
void set_log_full_action(in LogFullAction action);

Preconditions

- All the Domain profile files are available
- The source code files are available.

Test Description

A. Determine whether log services are provided. (OE0700, OE0700-C187, OE0700-C188, OE0700-C189, OE0700-C190, OE0700-C191)
   1. **Pass**: Log service is provided.
   2. **Untested:** No log service is provided.

B. For each log service identified in Step A, perform the following steps:
   1. Validate that the *get_log_full_action* operation returns the LogFullAction with a value of HALT or WRAP. (OE0700-C187, OE0700-C188)
      a. **Pass:** The LogFullAction is returned and the value is HALT or WRAP
      b. **Fail:** The LogFullAction is not returned.
      c. **Fail:** The value of LogFullAction is not HALT or WRAP.
   2. Validate that *set_log_full_action* operation will only accept the values of HALT or WRAP and that it sets LogFullAction to the valid input value. (OE0700-C189, OE0700-C190)
      a. **Pass:** Input values other than HALT or WRAP are rejected and the LogFullAction is set to the valid input value.
      b. **Fail:** LogFullAction is set to a value other than WRAP or HALT.
   3. Validate that when the LogFullAction is set to WRAP in the *set_log_full_action* operation that the availability status logFull is set to false. (OE0700-C191)
      a. **Pass:** The availability status logFull is set to false.
      b. **Fail:** The availability status logFull is not set to false

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, N/A, or any other as appropriate.
2. The Test Recording Log is intended to record long comments and lists of SPD files, SCD files, PRF, and other file/data.

OE_TC_007				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination. (OE0700, OE0700-C187, OE0700-C188, OE0700-C189, OE0700-C190, OE0700-C191)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B.1 Verify that the *get_log_full_action* operation returns a *LogFullAction* with a value of HALT or WRAP. (OE0700-C187, OE0700-C188)				
1. Locate the *get_log_full_action* operation in the source code.	**Pass:** The operation *get_log_full_action* is found in the source code. (OE0700-C187, OE0700-C188) **Fail:** The operation *get_log_full_action* is not found in the source code. (OE0700-C187, OE0700-C188)		Failure of criterion OE0700-C187 or OE0700-C188 means failure of the requirement OE0700.	
2. Examine the source code and verify that the *get_log_full_action* operation returns a *LogFullAction*.	**Pass:** The LogFullAction is returned and the value is HALT or WRAP. (OE0700-C187, OE0700-C188) **Fail:** The LogFullAction is not returned (OE0700-C187). **Fail:** The value of LogFullAction is not HALT or WRAP. (OE0700-C188)		Failure of criterion OE0700-C187 or OE0700-C188 means failure of the requirement OE0700	
B.2 Validate that set_log_full_action operation will only accept the values of HALT or WRAP and that it sets LogFullAction to the valid input value. (OE0700-C189, OE0700-C190)				

OE_TC_007				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Locate the *set_log_full_action* operation in the source code.	**Pass:** The operation *set_log_full_action* is found in the source code. (OE0700-C189, OE0700-C190) **Fail:** The operation *set_log_full_action* is not found in the source code. (OE0700-C189, OE0700-C190)		Failure of criterion OE0700-C189 or OE0700-C190 means failure of the requirement OE0700	
4. Examine the source code and verify that the *set_log_full_action* operation returns a *LogFullAction*.	**Pass:** Input values other than HALT or WRAP are rejected. (OE0700-C190) **Fail:** Input values other than HALT or WRAP are accepted (not rejected). (OE0700-C190) **Pass:** LogFullAction is set to the valid input value. (OE0700-C189) **Fail:** LogFullAction is set to a value other than WRAP or HALT. (OE0700-C189)		Failure of criterion OE0700-C189 or OE0700-C190 means failure of the requirement OE0700	
B.3 Validate that when the LogFullAction is set to WRAP in the set_log_full_action operation that the availability status logFull is set to false. (OE0700-C191)				
5. Examine the source code and verify that when the LogFullAction is set to WRAP in the *set_log_full_action* operation that the availability status logFull is set to false.	**Pass:** The availability status logFull is set to false. (OE0700-C191) **Fail:** The availability status logFull is not set to false. (OE0700-C191)		Failure of criterion OE0700-C191 means failure of the requirement OE0700	
End of Test				

Test Recording Log - OE_TC_007				
DMD File:				
OE_TC_005, **Step 2** (Log Services)				
OE_TC_005, Step 4 (componentfileref refid)				
OE_TC_005, **Step 5** (SPD Files)				
OE_TC_005, **Step 6** (Executable file)				
Step 1 (*get_log_full_action* – Y/N?)	**Step 2** (*set_log_full_action* – Y/N?)	**Step 3** (LogFullAction returned – Y/N?)	**Step 4** (LogFullAction valid – Y/N?)	**Step 5** (logFull set to false – Y/N?)

Test Summary: OE_TC_007

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C187 _____

OE0700-C188 _____

OE0700-C189 _____

OE0700-C190 _____

OE0700-C191 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.4. OE_TC_008 - OMG Lightweight Log Service :: get_availability_status

Test Case Number: OE_TC_008

OMG Lightweight Log Service :: get_availability_status

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Service
OE0700-C192	The returned instance of the AvailabilityStatus type contains two Boolean values: off_duty, which indicates the log is disabled when true; and log_full, which indicates that all free space is depleted in the log storage area.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2, 05/15/2006	Page 3-2, Section 3.1.2.2
OMG Lightweight Log Service Specification	Version 1.1, 02/02/2005	Pages 2-11 and 3-11 to 3-12

Test Objective

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.

The two Boolean values are independent, so there are four possible states of the log service:

	off_duty == FALSE	**off_duty == TRUE**
log_full == FALSE	Log working normally	Log is disabled
log_full == TRUE	Log memory is full	Log is disabled AND Log memory is full

Places to Verify

Core Framework Log Service

OMG Log Interfaces

Data

```
struct AvailabilityStatus{boolean off_duty;
                          boolean log_full;};
```

Operations

AvailabilityStatus get_availability_status();

Preconditions

- All Domain Profile files and source code files are available.

Test Description

A. Determine whether log services are provided (OE0700, OE0700-C192).
   1. **Pass**: Log service is provided.
   2. **Untested:** No log service is provided.

B. For each log service identified in Step A, perform the following steps:
   1. Verify that the get_availability_status operation exists. (OE0700-C192).
      a. **Fail:** get_availability_status operation does not exist.
   2. Verify that get_availability_status operation returns the AvailabilityStatus, with two Boolean values indicating whether the log service is off_duty and whether the log service is log_full (OE0700-C192).
      a. **Pass:** A compliant AvailabilityStatus is returned.
      b. **Fail:** AvailabilityStatus is not returned.
      c. **Fail:** An AvailabilityStatus with some other format is returned.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log sheet is intended to record data for each step that requires recording of data.

OE_TC_008				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700, OE0700-C192)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
For each log service identified, perform the following steps.				
B.1: Verify that the get_availability_status function exists. (OE0700-C192)				
1. Identify the source code for get_availability_status operation.	**Pass**: The get_availability_status operation is present. (OE0700-C192) **Fail**: The get_availability_status operation is not present. (OE0700-C192)		Failure of this criterion OE0700-C192 means failure of the requirement OE0700.	
B.2: Verify that get_availability_status operation returns the AvailabilityStatus, with two Boolean values indicating whether the log service is off_duty and whether the log service is log_full. (OE0700-C192)				
2. Verify that get_availability_status returns AvailabilityStatus.	**Pass**: AvailabilityStatus is returned. (OE0700-C192) **Fail**: AvailabilityStatus is not returned. (OE-0700-C192)		struct AvailabilityStatus{ boolean off_duty; boolean log_full;}; Failure of this criterion OE0700-C192 means failure of the requirement OE0700.	

OE_TC_008				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that AvailabilityStatus contains a Boolean value indicating whether the log service is off or on duty (off_duty).	**Pass**: AvailabilityStatus contains a Boolean value indicating whether the log service is off or on. (OE0700-C192) **Fail**: AvailabilityStatus does not contain a Boolean value indicating whether the log service is off or on. (OE0700-C192)		Failure of this criterion OE0700-C192 means failure of the requirement OE0700.	
4. Verify that AvailabilityStatus contains a Boolean value indicating whether the log still has room for additional log entries or is full (log_full).	**Pass**: AvailabilityStatus contains a Boolean value indicating whether the log still has room for additional log entries or is full. (OE0700-C192) **Fail**: AvailabilityStatus does not contain a Boolean value indicating whether the log still has room for additional log entries or is full. (OE0700-C192)		Failure of this criterion OE0700-C192 means failure of the requirement OE0700.	
End of Test				

Test Recording Log – OE_TC_008			
DMD file			
OE_TC_005, Step 2 (log service)			
DCD file			
OE_TC_005, Steps 4 & 5 (log service)			
SPD file			
Step 1 (get_availability_status operation)	Step 2 (AvailabilityStatus)	Step 3 (off_duty status)	Step 4 (log_full status)

Test Summary: OE_TC_008

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

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

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

OE0700 _____

OE0700-C192 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.5. OE_TC_009 - OMG Lightweight Log Service :: administrative_state

Test Case Number: OE_TC_009

OMG Lightweight Log Service::administrative_state

Requirements

SCA v2.2.2 Tag	SCA v2.2.2
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification.
	OMG Lightweight
OE0700-C193	The *get_administrative_state* is used to read the administrative state of the Log.
OE0700-C194	If the state is locked, no new records are accepted. Reading of already stored records is not affected.
OE0700-C195	This operation allows one to affect the ability of the logging service to accept and store new logging records by administrative action.
OE0700-C196	If the state is locked, no new records are accepted.
OE0700-C197	Reading of already stored records is not affected.
OE0700-C198	If the state is set to unlocked, the log operates normally.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2, 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Page 3-12, Section 3.3.1.6 Page 3-24, Section 3.3.4.3

Test Objective

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. The objective of this test is to verify that the *get_administrative_state*() operation returns the action set by the *set_administrative_state()* operation.

Places to Verify

Core Framework Log Service

OMG log Interfaces

Data

enum AdministrativeState {locked, unlocked};

Operations

AdministrativeState get_administrative_state();
void set_administrative_state(in AdministrativeState state);

Preconditions

- All Domain Profile files and source code files are available.

Test Description

A. Determine from the domain profile whether log services are provided (OE0700).
   1. **Pass**: Log service is provided.
   2. **Untested:** No log service is provided.

B. For each log service, perform the following steps:
   1. Validate that *get_administrative_state*() and *set_administrative_state*() operations exist. (OE0700-C193, OE0700-C194, OE0700-C195, OE0700-C196, OE0700-C197, OE0700-C198)
      a. **Pass:** *get_administrative_state*() operation does exist (OE0700-C193, OE0700-C194).
      b. **Fail:** *get_administrative_state*() operation does not exist (OE0700-C193, OE0700-C194).
      c. **Pass:** *set_administrative_state*() operation does exist. (OE0700-C195, OE0700-C196, OE0700-C197, OE0700-C198)
      d. **Fail:** *set_administrative_state*() operation does not exist. (OE0700-C195, OE0700-C196, OE0700-C197, OE0700-C198)
   2. Validate that the *set_ administrative_state*() operation will set the AdministrativeState value input. (OE0700-C195)
      a. **Pass:** The proper AdministrativeState is set.
      b. **Fail:** The wrong AdministrativeState is set.
   3. Validate that the *get_administrative_state*() operation returns the proper value. (OE0700-C193)
      a. **Pass:** The proper AdministrativeState is returned.
      b. **Fail:** The wrong AdministrativeState is returned.
   4. Verify that the write_records() and the write_record() operation checks the AdministrativeState and does not write records when the state is locked, but does write records when the state is unlocked(OE0700-C194, OE0700-C196).
      a. **Pass:** Records are written when the AdministrativeState is unlocked and are not written when the state is locked.
      b. **Fail:** Records are written when the AdministrativeState is locked
      c. **Fail:** Records are not written when the AdministrativeState is unlocked.
   5. Verify that the retrieve_records and retrieve_record operation will perform properly regardless of the value of the AdministrativeState value(OE0700-C194, OE0700-C197).
      a. **Pass:** Records are retrieved regardless of the value of AdministrativeState.

b. **Fail:** Records are not retrieved if the AdministrativeState is locked.
c. **Fail:** Records are not retrieved if the AdministrativeState is unlocked.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, N/A, or any other as appropriate.
2. The Test Recording Log is intended to record long comments and lists other file/data.

OE_TC_009				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (OE0700).				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B. For each log service, perform the following steps:				
B.1. Validate that *get_administrative_state*() and *set_administrative_state*() operations exist. (OE0700-C193, OE0700-C194, OE0700-C195, OE0700-C196, OE0700-C197, OE0700-C198)				
1. Locate the source code for the *get_administrative_state*() operation.	**Pass:** The *get_administrative_state*() operation is found. (OE0700-C193, OE0700-C194) **Fail:** The *get_administrative_state*() operation is not found. (OE0700-C193, OE0700-C194)		Failure of criterion OE0700-C193 and OE0700-C194 means failure to the requirement OE0700.	
2. Locate the source code for the *set_administrative_state*() operation.	**Pass:** The *set_administrative_state*() operation is found. (OE0700-C195, OE0700-C196, OE0700-C197 and OE0700-C198) **Fail:** The *set_administrative_state*() operation is found. (OE0700-C195, OE0700-C196, OE0700-C197 and OE0700-C198)		Failure of criterion OE0700-C195, OE0700-C196, OE0700-C197 and OE0700-C198 means failure to the requirement OE0700.	
B.2. Validate that the *set_ administrative_state*() operation will set the AdministrativeState value input. (OE0700-C195)				
3. Validate that the set_ administrative_state() operation will set the AdministrativeState the value that was input.	**Pass:** AdministrativeState is set to the input value. (OE0700-C195) **Fail:** AdministrativeState is not set to the input value. (OE0700-C195)		Failure of criterion OE0700-C195 means failure to the requirement OE0700.	

OE_TC_009				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
B.3. Validate that the *get_administrative_state*() operation returns the proper value. (OE0700-C193)				
4. Validate that the *get_administrative_state*() operation returns the proper value.	**Pass:** The value of the AdministrativeState is returned. (OE0700-C193) **Fail:** The value of the AdministrativeState is not returned. (OE0700-C193)		Failure of criterion OE0700-C193 means failure to the requirement OE0700.	
B.4. Verify that the write_records() and the write_record() operation checks the AdministrativeState and does not write records when the state is locked, but does write records when the state is unlocked. (OE0700-C194, OE0700-C196)				
5. Verify that the write_record() operation checks the AdministrativeState and does not write records when the state is locked.	**Pass:** The write_record() operation does not write records when the AdministrativeState is locked. (OE0700-C194, OE0700-C196) **Fail:** The write_record() operation writes records when the AdministrativeState is locked. (OE0700-C194, OE0700-C196)		Failure of criterion OE0700-C194 and OE0700-C196 means failure to the requirement OE0700.	
6. Verify that the write_records() operation checks the AdministrativeState and does not write records when the state is locked.	**Pass:** The write_records() operation does not write records when the AdministrativeState is locked. (OE0700-C194, OE0700-C196) **Fail:** The write_records() operation writes records when the AdministrativeState is locked. (OE0700-C194, OE0700-C196)		Failure of criterion OE0700-C194 and OE0700-C196 means failure to the requirement OE0700.	
B.5. Verify that the retrieve_records and retrieve_record operation will perform properly regardless of the value of the AdministrativeState value. (OE0700-C194, OE0700-C197)				

OE_TC_009				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
7. Verify that the retrieve_record() operation will perform properly regardless of the value of the AdministrativeState value.	**Pass:** The retrieve_record() operation performs properly. (OE0700-C194, OE0700-C197) **Fail:** The retrieve_record() operation does not perform properly. (OE0700-C194, OE0700-C197)		Failure of criterion OE0700-C194 and OE0700-C197 means failure to the requirement OE0700.	
8. Verify that the retrieve_records() operation will perform properly regardless of the value of the AdministrativeState value.	**Pass:** The retrieve_records() operation performs properly. (OE0700-C194, OE0700-C197) **Fail:** The retrieve_records() operation does not perform properly. (OE0700-C194, OE0700-C197)		Failure of criterion OE0700-C194 and OE0700-C197 means failure to the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_009			
DMD file			
OE_TC_005, Step 2 (log service)			
DCD file			
OE_TC_005, Step 4 (log service – *componentplacement)*	OE_TC_005, Step 5 (log service - *componentfile id)*		
SPD file			
Step 1 (*get_administrative_state* and *set_administrative_state* operations)	Step 2 & 3 (set_administrative_state operation)	Step 4 (*get_administrative_state* operation)	Steps 5 & 6 (write_record operation)
Step 7 & 8 (retrieve_record operation)			

Test Summary: OE_TC_009

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C193 _____

OE0700-C194 _____

OE0700-C195 _____

OE0700-C196 _____

OE0700-C197 _____

OE0700-C198 _____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.6. OE_TC_010 - OMG Lightweight Log Service :: get_operational_state

Test Case Number: OE_TC_010

OMG Lightweight Log Service::get_operational_state

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Specification
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification (OE0700-C199 – OE0700-C200 list below).
	OMG Lightweight Log Specification
OE0700-C199	The *get_operational_state()* operation returns the actual operational state of the log.
OE0700-C200	Possible values are ENABLED, which means the log is fully functional and available to log producer and log consumer clients; or DISABLED, which indicates the log has encountered a runtime problem and is not available for use by log producers or log consumers.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Section 3.2.3 page. 3-3, Section 3.3.1.7, page. 3-13

Test Objective

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C199 and OE0700-C200 of the OMG Lightweight Log. 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.

Places to Verify

Core Framework Log Service

OMG Log Interfaces

Data

enum OperationalState {disabled, enabled};

Operations

OperationalState getOperationalState ()

Preconditions

- All the Domain Profile files are available.
- The source code files are available.

Test Description

A. Determine from the domain profile whether log services are provided. (OE0700)
   1. **Pass**: Log service is provided.
   2. **Untested**: No log services are provided.

B. Verify that the *get_operational_state()* operation exists and returns the actual operational state of the log. (OE0700-C199).
   1. **Fail**: The *get_operational_state()* does not correctly return the actual operational state of the log.
   2. **Pass**: The get_operational_state() returns the actual operational state of the log
C. Verify that the *get_operational_state()* function returns the enumerated type, *OperationalState*, with a value of ENABLED or DISABLED. (OE0700-C200).
   1. **Fail**: The *get_operational_state()* does not return a value of either ENABLED or DISABLED.
   2. **Pass**: The *get_operational_state()* returns a value of ENABLED or DISABLED that accurately indicates the operational state of the system. If the *get_operational_state()* returns ENABLED, it should be available to both producers and consumers.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested or N/A.
2. The Test Recording Log sheet is intended to record data from the test steps.

OE_TC_010				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination). (OE0700)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B. Verify that the get_operational_state() operation exists and returns the actual operational state of the log. (OE0700-C199)				
1. Examine the source code and verify that the *get_operational_state()* operation exists.	**Pass**: The *get_operational_state()* operation exists. (OE0700-C199) **Fail**: The *get_operational_state()* operation does not exist. (OE0700-C199)		Failure of this criterion OE0700-C199 means failure of the requirement OE0700.	
2. Examine the source code and verify that the *get_operational_state()* operation returns the actual operational state of the log.	**Pass**: The *get_operational_state()* operation correctly returns the actual operational state of the log. (OE0700-C199) **Fail**: The *get_operational_state()* operation does not correctly return the actual operational state of the log. (OE0700-C199)		**Operation**alState <<enumeration>> ENABLED DISABLED Failure of this criterion OE0700-C199 means failure of the requirement OE0700.	
C. Verify that the *get_operational_state()* function returns the enumerated type, *OperationalState*, with a value of ENABLED or DISABLED. This status must accurately indicate the availability of the log to producers and consumers. (OE0700-C200)				

OE_TC_010				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Examine the source code and verify that the instance of the enumerated type returned by *get_operational_state()* is either ENABLED, indicating that the log is fully functional, or DISABLED, indicated that the log is unavailable to either producers or consumers.	**Pass**: The get_operational_state() returns a proper value to indicate the operational state of the log service. (OE0700-C200) **Fail**: The get_operational_state() does not return a proper value to indicate the operational state of the log service. (OE0700-C200)		The write_records() operation, as well as other Log operations, should not be allowed access to the log when the *get_operational_state()* returns a DISABLED status. Failure of this criterion OE0700-C200 means failure of the requirement OE0700.	
End of Test				

Test Recording Log OE_TC_010					
DMD File					
DCD File					
OE_TC_005, Step 1-5 (Log service)	OE_TC_005, Step 6 (Executable name)	Step 1 *(get_operational_state()* exists)	Step 2 *(get_operational_state()* returns the actual operational state of the log)	Step 3	
				(Returns an *OperationalState* enum with value of either ENABLED or DISABLED)	(OperationalState value accurately reflects the operational state of the system.)

Test Summary: OE_TC_010

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C199 _____

OE0700-C200 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.7. OE_TC_011 - OMG Lightweight Log Service :: get_record_id_from_time

Test Case Number: OE_TC_011

OMG Lightweight Log Service::get_record_id_from_time

Requirements

SCA v2.2.2 Tag	SCA v2.2.2
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification.
	OMG Lightweight
OE0700-C201	The *get_record_id_from_time operation* returns the record Id of the first record in the Log with a time stamp that is greater than, or equal to, the time specified in the fromTime parameter.
OE0700-C202	If the Log does not contain a record that meets the criteria provided, then the RecordId returned corresponds to the next record that will be recorded in the future.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Page 3-14, section 3.3.2.1

Test Objective

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.

Places to Verify

Log Services

OMG Log Interfaces

Data

```
typedef unsigned long long RecordId;
struct LogTime {long seconds;
                long nanoseconds; };
```

Operations

RecordId get_record_id_from_time (in LogTime fromTime);

Preconditions

- All the Domain profile files are available.
- The source code files are available.

Test Description

A. Determine whether log services are provided. (OE0700, OE0700-C201, OE0700-C202)
   1. **Pass**: Log service is provided.
   2. **Untested:** No log service is provided.

B. For each log service identified in Step A, perform the following steps:
   1. Validate that the *get_record_id_from_time* operation returns the RecordId of the first log record with a time stamp greater than or equal to the requested time. (OE0700-C201)
      a. **Pass:** The proper RecordId is returned.
      b. **Fail:** The proper RecordId is not returned.
   2. Validate that if no record meets the time criteria, then the RecordId for the next record to be recorded is returned. (OE0700-C202)
      a. **Pass:** The proper RecordId is returned.
      b. **Fail:** The correct RecordId is not returned.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, N/A, or any other as appropriate.
2. The Test Recording Log is intended to record long comments and lists of SPD files, SCD files, PRF, and other file/data.

OE_TC_011				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700, OE0700-C201, OE0700-C202)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B.1. For each log service identified in Step A, Validate that the *get_record_id_from_time* operation returns the RecordId of the first log record with a time stamp greater than or equal to the requested time. (OE0700-C201, OE0700-C202)				
1. Locate the *get_record_id_from_time* operation in the source code.	**Pass**: The operation is found in the source code. (OE0700-C201, OE0700-C202) **Fail**: The operation is not found in the source code. (OE0700-C201, OE0700-C202)		Failure of these criteria OE0700-C201 and OE0700-C202 means failure of the requirement OE0700.	
2. Verify that the get_*record_id_from_time* will find the first RecordId whose time is greater than or equal to the fromTime parameter and returns this record.	**Pass**: The proper RecordId is returned (OE0700-C201). **Fail**: The proper RecordId is not returned (OE0700-C201)		Failure of this criterion OE0700-C201 means failure of the requirement OE0700.	
B.2 For each log service identified in Step A, Validate that if no record meets the time criteria, then the RecordId for the next record to be recorded is returned				
3. Verify that, if no record meets the time requirement, the RecordId of the next record to be written is retumed.	**Pass**: The correct RecordId is returned (OE0700-C202). **Fail**: The correct RecordId is not returned (OE0700-C202)		Failure of this criterion OE0700-C202 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_011							
DMD file							
DCD file							
OE_TC_005, Step 2 (logging service)	OE_TC_005, Step 4 (*componentfileref*)	OE_TC_005, Step 5 (SPD file)	OE_TC_005, Step 6 (local file name)	Step 1 (get_record_id_from_time)	Step 2 (record exists)	Step 3 (record does not exist)	Notes

Test Summary: OE_TC_011

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s)(x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C201 _____

OE0700-C202 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.8. OE_TC_012 - OMG Lightweight Log Service :: retrieve_records

Test Case Number: OE_TC_012

OMG Lightweight Log Service::retrieve_records

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Specification
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification.
	OMG Lightweight Log Specification
OE0700-C203	The *retrieve_records* operation returns a *LogRecordSequence* that begins with the record specified by the *currentId* parameter.
OE0700-C204	The number of records in the *LogRecordSequence* returned by the *retrieve_records* operation is equal to the number of records specified by the *howMany* parameter, or the number of records available if the number of records specified by the *howMany* parameter cannot be met.
OE0700-C205	The log will update *howMany* to indicate the number of records returned and will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.
OE0700-C206	If the record specified by *currentId* does not exist, but corresponds to the next record that will be recorded in the future, the *retrieve_records* operation returns an empty list of LogRecords, sets *howMany* to zero, and leaves the value of *currentId* unchanged.
OE0700-C207	If the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, the *retrieve_records* operation returns an empty list of LogRecords, and sets both, *currentId* and *howMany* to zero.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2, section 3.1.2.2
Lightweight Log Service Specification	Version 1.1 Feb 2005	Page 3-15, section 3.3.2.2

Test Objective

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. 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.

Places to Verify

Core Framework Log Service

OMG Log Interfaces

Data

```
typedef unsigned long long RecordId;
struct LogTime {long seconds;
                long nanoseconds; };
typedef unsigned short LogLevel;
struct ProducerLogRecord {    string producerId;
                              string producerName;
                              LogLevel level;
                              string logData; };
struct LogRecord {      RecordId id;
                        LogTime time;
                        ProducerLogRecord info; };

typedef sequence<LogRecord> LogRecordSequence;
```

Operations

```
LogRecordSequence retrieve_records( inout RecordId currentId,
                                    inout unsigned long howMany);
```

Preconditions

- All the Domain profile files are available.
- The source code files are available.

Test Description

A. Determine from the domain profile whether log services are provided (OE0700).
   1. **Pass**: Log service is provided.
   2. **Untested:** No log service is provided.

B. For each log service, perform the following steps:
   1. Verify that the *retrieve_records* operation returns a *LogRecordSequence* that begins with the record specified by the *currentId* parameter (OE0700-C203)
      a. **Pass:** The *LogRecordSequence* and *currentId* point to the same record.
      b. **Fail:** The *LogRecordSequence* and *currentId* do not point to the same record.
   2. Verify that the number of records returned in the *LogRecordSequence* is equal to or less than the number of records specified by the *howMany* parameter (OE0700-C204).
      a. **Pass:** The number of records in the *LogRecordSequence* returned is equal to or less than the number of records specified by the *howMany* parameter
      b. **Fail:** The number of records in the *LogRecordSequence* returned is greater than the number of records specified by the *howMany* parameter.
   3. Verify that the log will update *howMany* to indicate the number of records returned, and set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available. (OE0700-C205)
      a. **Pass:**
         (1) The *howMany* indicates the number of records returned, and
         (2) sets the *currentId* to either the id of the next record or the id of next record that will be recorded in the future if there are no further records available.
      b. **Fail:**
         (1) The *howMany* does not indicate the number of records returned or
         (2) does not set the *currentId* to either the id of the next record or the id of next record that will be recorded in the future if there are no further records available.

C. This item will be tested IF the record specified by *currentId* does not exist, but corresponds to the next record that will be recorded in the future. Verify that the *retrieve_records* operation returns an empty list of LogRecords, and sets *howMany* to zero, and leaves the value of *currentId* unchanged (OE0700-C206).
   1. **Pass:** The *retrieve_records* operation
      a. returns an empty list of LogRecords, and
      b. sets *howMany* to zero, and
      c. leaves the value of *currentId* unchanged.

2. **Fail:** The *retrieve_records* operation
   a. does not return an empty list of LogRecords, or
   b. does not set *howMany* to zero, or
   c. does not leave the value of *currentId* unchanged.

D. This item will be test IF the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty. Verify that the *retrieve_records* operation returns an empty list of LogRecords, and sets both *currentId* and *howMany* to zero (OE0700-C207).
   1. **Pass:** The *retrieve_records* operation
      a. returns an empty list of LogRecords and
      b. sets both *currentId* and *howMany* to zero
   2. **Fail:** The *retrieve_records* operation
      a. does not return an empty list of LogRecords, or does not set both *currentId* and *howMany* to zero

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, N/A, or any other as appropriate.
2. The Test Recording Log sheet is intended to record data for each step that requires recording of data.

OE_TC_012				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination). (OE0700)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
For each log service, perform the following steps: C.1, C.2 and C.3				
B.1. Verify that the *retrieve_records* operation returns a *LogRecordSequence* that begins with the records pecified by the *currentId* parameter. (OE0700-C203)				
1. Examine the source code and verify that the *retrieve_records* operation returns a *LogRecordSequence* that begins with the record specified by the *currentId* parameter.	**Pass***: retrieve_records* operation returns a *LogRecordSequence* that begins with the record specified by the *currentId* parameter that shows that the *LogRecordSequence and currentId point to the same record*. (OE0700-C203) **Fail***: retrieve_records* operation does not return a *LogRecordSequence* that begins with the record specified by the *currentId* parameter. (OE0700-C203)		Failure of this criterion OE0700-C203 means failure of the requirement OE0700.	
B.2. Verify that the number of records returned in the *LogRecordSequence* is equal to or less than the number of records specified by the *howMany* parameter. (OE0700-C204)				

OE_TC_012				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Verify that the number of records in the *LogRecordSequence* returned by the *retrieve_records* operation • is equal to the number of records specified by the *howMany* parameter or • is less than the number of records specified by the *howMany* parameter, if the number of records specified by the howMany parameter cannot be met.	**Pass**: The number of records in the *LogRecordSequence* returned by the *retrieve_records* operation • is equal to the number of records specified by the *howMany* parameter or • is less than the number of records specified by the *howMany* parameter, if the number of records specified by the howMany parameter cannot be met. (OE0700-C204) **Fail**: The number of records in the *LogRecordSequence* returned by the *retrieve_records* operation is greater than the number of records specified by the *howMany* parameter. (OE0700-C204)		Failure of this criterion OE0700-C204 means failure of the requirement OE0700.	
B.3. Verify that the log will update *howMany*. (OE0700-C205)				

OE_TC_012				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Examine the source code and verify the *retrieve_records* operation that - the log will update *howMany* to indicate the number of records returned - and will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.	**Pass**: - the log will update *howMany* to indicate the number of records returned. and - the log will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available. (OE0700-C205) **Fail**: - the log does not indicate the number of records returned or - the log will not set *currentId* to either the id of the next record or the id of the next record that will be recorded in the future if there are no further records available. (OE0700-C205)		Failure of this criterion OE0700-C205 means failure of the requirement OE0700.	
C. Verify that the retrieve_records operation returns an empty list of LogRecords, sets howMany to zero and leaves the value of *currentId* unchanged. (OE0700-C206)				

OE_TC_012				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that the retrieve_records operation • returns an empty list of LogRecords, • sets howMany to zero, and • leaves the value of currentId unchanged, if any of the following conditions are true: a. if the record specified by currentId does not exist or b. if the currentId does correspond to the next record that will be recorded in the future.	**Pass**: The *retrieve_records* operation • returns an empty list of LogRecords, • sets *howMany* to zero, and • leaves the value of *currentId* unchanged if the record specified by *currentId* does not exist or corresponds to the next record that will be recorded in the future. (OE0700-C206) **Fail**: The *retrieve_records* operation • does not return an empty list of LogRecords, • does not set howMany to zero, or • does not leave the value of currentId unchanged if the record specified by *currentId* does not exist but corresponds to the next record that will be recorded in the future. (OE0700-C206)		Failure of this criterion OE0700-C206 means failure of the requirement OE0700.	
D. Verify that the *retrieve_records* operation returns an empty list of LogRecords and sets both *currentId* and *howMany* to zero. (OE0700-C207)				

OE_TC_012				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
5. Verify that the retrieve_records operation • returns an empty list of LogRecords, • sets both, currentId and • sets howMany to zero, if either of the following conditions are true: a. if the record specified by currentId does not exist and if the currentId does not correspond to the next record that will be recorded in the future, or b. if the Log is empty.	**Pass**: The *retrieve_records* operation • returns an empty list of LogRecords, and • sets currentId to zero and • sets howMany to zero if the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty. (OE0700-C207) **Fail**: The *retrieve_records* operation • does not return an empty list of LogRecords, • does not set currentId and • does not set howMany to zero if the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty). (OE0700-C207)		Failure of this criterion OE0700-C207 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_012							
(Step 1) DMD File:							
(Step 3) DCD File:							
Step 2 Log Services:							
OE_TC_005, **Step 4 (componentfileref refid)**	OE_TC_005, **Step 5 (SPD Files)**	OE_TC_005, **Step 6 (Executable Source Files)**	**Step 1 (retrieve_record)**	**Step 2 (Less Than or Equal)**	**Step 3 (howMany)**	**Step 4 (currentId corresponds)**	**Step 5 (currentId not corresponds)**

Test Summary: OE_TC_012

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

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

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

OE0700 _____

OE0700-C203 _____

OE0700-C204 _____

OE0700-C205 _____

OE0700-C206 _____

OE0700-C207 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.9. OE_TC_013 - OMG Lightweight Log Service :: retrieve_records_by_level

Test Case Number: OE_TC_013

OMG Lightweight Log Service::retrieve_records_by_level

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Specification
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Specification
OE0700-C208	The retrieve_records_by_level operation returns a *LogRecordSequence* of records that correspond to the supplied *LogLevels*.
OE0700-C209	Candidate records for the *LogRecordSequence* begin with the record specified by the *currentId* parameter.
OE0700-C210	The number of records in the *LogRecordSequence* returned by the *retrieve_records_by_level* operation is equal to the number of records specified by the *howMany* parameter, or the number of records available if the number of records specified by the *howMany* parameter cannot be met.
OE0700-C211	The log will update *howMany* to indicate the number of records returned and will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.
OE0700-C212	If the record specified by *currentId* does not exist, but corresponds to the next record that will be recorded in the future, the *retrieve_records_by_level* operation returns an empty list of LogRecords, sets *howMany* to zero, and leaves the value of *currentId* unchanged.
OE0700-C213	If the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, the *retrieve_records_by_level* operation returns an empty list of LogRecords, and sets both, *currentId* and *howMany* to zero.

References

Document Name	Version/Date	Pages referenced
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Page 3-15, Section 3.3.2.2

Test Objective

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. 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.

Places to Verify

Log Service

OMG Log Interfaces

Data

```
typedef unsigned long long RecordId;
typedef unsigned short LogLevel;
typedef sequence<LogLevel> LogLevelSequence;
struct LogTime {long seconds;
                long nanoseconds; };
struct ProducerLogRecord {string producerId;
                          string producerName;
                          LogLevel level;
                          string logData; };
struct LogRecord {RecordId id;
                  LogTime time;
                  ProducerLogRecord info; };
typedef sequence<LogRecord> LogRecordSequence;
```

Operations

```
LogRecordSequence retrieve_records_by_level(
                                    inout RecordId currentId,
                                    inout unsigned long howMany,
                                    in LogLevelSequence valueList);
```

Preconditions

- All the Domain profile files are available
- The source code files are available.

Test Description

A. Determine from the domain profile whether log services are provided. (OE0700)
   1. **Pass**: Log service is provided
   2. **Untested:** No log service is provided.

For each log service, perform the following steps:

B. Verify that retrieve_records_by_level operation returns a *LogRecordSequence* of records that correspond to the supplied *LogLevelSequence* (OE0700-C208).
   1. **Pass:** The *LogRecordSequence* and the supplied *LogLevelSequence* point to the same log levels.
   2. **Fail:** The *LogRecordSequence* and the supplied *LogLevelSequence* do not point to the same log levels.

C. Verify that the candidate records for the LogRecordSequence begin with the record specified by the currentID parameter (OE0700-C209).
   1. **Pass:** The candidate records for the LogRecordSequence begin with the record specified by the currentID parameter
   2. **Fail:** The candidate records for the LogRecordSequence do not begin with the record specified by the currentID parameter.

D. Verify that the number of records in the *LogRecordSequence* returned is equal to or less than the number of records specified by the *howMany* parameter (OE0700-C210).
   1. **Pass:** The number of records in the *LogRecordSequence* returned is equal to or less than the number of records specified by the *howMany* parameter
   2. **Fail:** The number of records in the *LogRecordSequence* returned is greater than the number of records specified by the *howMany* parameter.

E. Verify that the log will update *howMany* (OE0700-C211).
      a. to indicate the number of records returned, and
      b. set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.
   1. **Pass:**
      a. The *howMany* indicates the number of records returned, and
      b. sets the *currentId* to either the id of the next record or the id of next record that will be recorded in the future if there are no further records available.
   2. **Fail:**
      a. The *howMany* does not indicate the number of records returned, or
      b. does not set the *currentId* to either the id of the next record or the id of next record that will be recorded in the future if there are no further records available.

F. Verify that IF the record specified by *currentId* does not exist, but corresponds to the next record that will be recorded in the future, then the *retrieve_records_by_level* operation…

- returns an empty list of LogRecords, and
- sets *howMany* to zero, and
- leaves the value of *currentId* unchanged. (OE0700-C212)

1. **Pass:** The *retrieve_records_by_level* operation
   - returns an empty list of LogRecords, and
   - sets *howMany* to zero, and
   - leaves the value of *currentId* unchanged.
2. **Fail:** The *retrieve_records_by_level* operation
   - does not return an empty list of LogRecords, or
   - does not set *howMany* to zero, or
   - does not leave the value of *currentId* unchanged.

G. Verify that IF the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, then the *retrieve_records_by_level* operation…
   - returns an empty list of LogRecords, and
   - sets both *currentId* and *howMany* to zero (OE0700-C213).

1. **Pass:** The *retrieve_records_by_level* operation
   - returns an empty list of LogRecords and
   - sets both *currentId* and *howMany* to zero.
2. **Fail:** The *retrieve_records_by_level* operation
   - does not return an empty list of LogRecords, or
   - does not set both *currentId* and *howMany* to zero.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, N/A, or any other as appropriate.
2. The Test Recording Log sheet is intended to record long comments and other file/data.

OE_TC_013				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination). (OE0700)				
If it has not already been done, perform steps 1 thru 6 of OE_TC_005 to determine if a Log Service is properly defined for the OE under test.				
B. Verify that retrieve_records_by_level operation returns a *LogRecordSequence* of records that correspond to the supplied *LogLevelSequence* (OE0700-C208).				
1. Examine the source code and verify that the *retrieve_records_by_level* operation returns a *LogRecordSequence* that corresponds to the supplied *LogLevelSequence*.	**Pass:** *retrieve_records_by_level* operation returns a *LogRecordSequence* that corresponds to the supplied *LogLevelSequence*. (OE0700-C208) **Fail:** *retrieve_records_by_level* operation does not return a *LogRecordSequence* that corresponds to the supplied *LogLevelSequence*. (OE0700-C208)		Failure of this criterion OE0700-C208 means failure of the requirement OE0700.	
C. Verify that the candidate records for the *LogRecordSequence* begin with the records specified by the *currentId* parameter (OE0700-C209).				
2. Examine the source code and verify the candidate records for the *LogRecordSequence* begin with the record specified by the *currentId* parameter.	**Pass:** The candidate records for the *LogRecordSequence* begin with the record specified by the *currentId* parameter. (OE0700-C209) **Fail:** The candidate records for the *LogRecordSequence* do not begin with the record specified by the *currentId* parameter. (OE0700-C209)		Failure of this criterion OE0700-C209 means failure of the requirement OE0700.	
D. Verify that the number of records in the *LogRecordSequence* returned is equal to or less than the number of records specified by the *howMany* parameter (OE0700-C210).				

OE_TC_013				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Examine the source code and verify the *retrieve_records_by_level* operation that the number of records in the *LogRecordSequence* returned by the *retrieve_records_by_level* operation is equal to or less the number of records specified by the *howMany* parameter.	**Pass:** The Number of records in the L*ogRecordSequence* returned by the *retrieve_records_by_level* operation is equal to or less than the number of records specified by the *howMany* parameter. (OE0700-C210) **Fail:** The Number of records in the L*ogRecordSequence* returned by the *retrieve_records_by_level* operation is greater than the number of records specified by the *howMany* parameter. (OE0700-C210)		Failure of this criterion OE0700-C210 means failure of the requirement OE0700.	
E. Verify that the log will update *howMany* (OE0700-C211).				
4. Examine the source code and verify the *retrieve_records_by_level* operation that - the log will update *howMany* to indicate the number of records returned - and will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.	**Pass:** The log will update *howMany* to indicate the number of records returned and the log will set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future, if there are no further records available. (OE0700-C211) **Fail:** The log does not update *howMany* to indicate the number of records returned and the log will not set *currentId* to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available. (OE0700-C211)		Failure of this criterion OE0700-C211 means failure of the requirement OE0700.	

OE_TC_013				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
F. Verify that IF the record specified by *currentId* does not exist, but corresponds to the next record that will be recorded in the future, then the *retrieve_records_by_level* operation… • **returns an empty list of LogRecords, and** • **sets *howMany* to zero, and** • **leaves the value of currentId unchanged. (OE0700-C212)**				
5. Examine the source code and verify that, if the record specified by *currentId* does not exist but corresponds to the next record that will be recorded in the future, then the *retrieve_records_by_level* operation • returns an empty list of LogRecords, • sets *howMany* to zero, and • leaves the value of *currentId* unchanged	**Pass:** If the record specified by *currentId* does not exist but corresponds to the next record that will be recorded in the future, then the *retrieve_records_by_level* operation • returns an empty list of LogRecords, • sets *howMany* to zero, and • leaves the value of *currentId* unchanged. (OE0700-C212) **Fail:** If the record specified by *currentId* does not exist but corresponds to the next record that will be recorded in the future, then the *retrieve_records_by_level* operation • does not return an empty list of LogRecords, • does not set *howMany* to zero, or • does not leave the value of *currentId* unchanged. (OE0700-C212)		Failure of this criterion OE0700-C212 means failure of the requirement OE0700.	
G. Verify that IF the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, then the *retrieve_records_by_level* operation… • **returns an empty list of LogRecords, and** • **sets both currentId and howMany to zero (OE0700-C213).**				

OE_TC_013				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Examine the source code and verify that if the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, then the *retrieve_records_by_level* operation • returns an empty list of LogRecords, and • sets both, *currentId* and *howMany* to zero.	**Pass:** If the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, then the *retrieve_records_by_level* operation • returns an empty list of LogRecords, and • sets both *currentId* and *howMany* to zero. (OE0700-C213) **Fail:** If the record specified by *currentId* does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, then the *retrieve_records_by_level* operation • does not return an empty list of LogRecords, and • does not set both *currentId* and *howMany* to zero. (OE0700-C213)		Failure of this criterion OE0700-C213 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_013			
DMD File:			
DCD File:			
Step2 (Log Services)	**Step3 (componentfileref refid)**	**Step5 (SPD Files)**	**Step6 (Executable Source Files)**

Test Summary: OE_TC_013

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C208 _____

OE0700-C209 _____

OE0700-C210 _____

OE0700-C211 _____

OE0700-C212 _____

OE0700-C213 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.10. OE_TC_014 - OMG Lightweight Log Service :: retrieve_records_by_producer_id

Test Case Number: OE_TC_014

OMG Lightweight Log Service::retrieve_records_by_producer_id

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification [7].
	OMG Lightweight Log Specification
OE0700-C214	The *retrieve_records_by_producer_id* operation returns a LogRecordSequence of records that correspond to the supplied producerIds.
OE0700-C215	Candidate records for the LogRecordSequence begin with the record specified by the currentId parameter.
OE0700-C216	The number of records in the LogRecordSequence returned by the *retrieve_records_by_producer_id* operation is equal to the number of records specified by the howMany parameter, or the number of records available if the number of records specified by the howMany parameter cannot be met.
OE0700-C217	The log will update howMany to indicate the number of records returned and will set currentId to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.
OE0700-C218	If the record specified by currentId does not exist, but corresponds to the next record that will be recorded in the future, the *retrieve_records_by_producer_id* operation returns an empty list of LogRecords, sets howMany to zero, and leaves the value of currentId unchanged.
OE0700-C219	If the record specified by currentId does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, the *retrieve_records_by_producer_id* operation returns an empty list of LogRecords, and sets both, currentId and howMany to zero.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-2
OMG Lightweight Log Service Specification	Version 1.1 05-02-2002 (February 2005)	Pages 2-15 to 2-16 and 3-17 to 3-18

Test Objective

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.

Places to Verify

Log Services

OMG Log Interfaces

Data

```
typedef unsigned long long RecordId;
struct LogTime {long seconds;
                long nanoseconds; };
struct ProducerLogRecord {string producerId;
                          string producerName;
                          LogLevel level;
                          string logData; };
struct LogRecord {RecordId id;
                  LogTime time;
                  ProducerLogRecord info; };
typedef sequence<LogRecord> LogRecordSequence;
```

Operations

```
LogRecordSequence retrieve_records_by_producer_id(inout RecordId currentId,
                                                   inout unsigned long howMany,
                                                   in StringSeq valueList);
```

Preconditions

- All of the domain profile files and Operating Environment (OE) source code files are available.

Test Description

A. Determine whether log services are provided. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)
   1. **Untested**: No log service is provided.

For each log service identified in Step A, perform the following steps:

B. Verify that the log service provides the *retrieve_records_by_producer_id* method. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219).
   1. **Fail:** The *retrieve_records_by_producer_id* method is not present.

C. Verify that the return value from *retrieve_records_by_producer_id* is of type LogRecordSequence and contains only records with a producer_id identical to the input/output currentId parameter. (OE0700-C214)

1. **Pass:** The *retrieve_records_by_producer_id* method returns a LogRecordSequence.
2. **Fail:** The *retrieve_records_by_producer_id* method does not return a LogRecordSequence.

D. Verify the use of the currentId parameter
   1. Verify that the input parameter currentId is used by the *retrieve_records_by_producer_id* method to determine which records are returned. If the currentId corresponds to an existing log record, then that log record will be first in the returned LogRecordSequence. (OE0700-C215)
      a. **Pass:** The returned LogRecordSequence will begin with currentId.
      b. **Fail:** The *retrieve_records_by_producer_id* method returns a (non-empty) LogRecordSequence that begins with a different record than currentId.
   2. Test for the case where the input parameter currentId corresponds to the next record to be written (in the future). Verify that the returned LogRecordSequence is empty, and that the returned parameter howMany is zero, and that the returned parameter currentId is unchanged from the input value currentId. (OE0700-C218)
      a. **Pass:** The *retrieve_records_by_producer_id* method returns an empty LogRecordSequence, zero for howMany, and an unchanged currentId.
      b. **Fail:** The *retrieve_records_by_producer_id* method does not return an empty LogRecordSequence.
      c. **Fail:** The *retrieve_records_by_producer_id* method returns a non-zero howMany.
      d. **Fail:** The *retrieve_records_by_producer_id* method returns a changed currentId.
   3. Test for the case where the input parameter currentId corresponds to some record that is beyond the next record to be written. Verify that the returned LogRecordSequence is empty, and that the returned parameter howMany is zero, and that the returned parameter currentId is zero. (OE0700-C219)
      a. **Pass:** The *retrieve_records_by_producer_id* method returns an empty LogRecordSequence, zero for howMany, and zero for currentId.
      b. **Fail:** The *retrieve_records_by_producer_id* method returns an empty LogRecordSequence, zero for howMany, and zero for currentId.
      c. **Fail:** The *retrieve_records_by_producer_id* method returns a non-zero howMany.
      d. **Fail:** The *retrieve_records_by_producer_id* method returns a non-zero currentId.

E. Verify the use of the howMany parameter.
   1. Verify that the input parameter howMany is used by the *retrieve_records_by_producer_id* method to determine the number of records returned, where there are **exactly** howMany records remaining in the log, beginning with currentID. (OE0700-C216)
      a. **Pass:** The returned LogRecordSequence will be of the length howMany, for the case where there are at least that many records remaining, (must also pass step E.2.a and E.3.a).
      b. **Fail:** The *retrieve_records_by_producer_id* method returns a LogRecordSequence with a length that is not equal to the parameter howMany .

2. Verify that the input parameter howMany is used by the *retrieve_records_by_producer_id* method to determine the number of records returned, where there are **more** than howMany records remaining in the log, beginning with currentID. (OE0700-C216)
   a. **Pass:** The returned LogRecordSequence will be of the length howMany, for the case where there are at least that many records remaining. (must also pass step E.1.a and E.3.a)
   b. **Fail:** The *retrieve_records_by_producer_id* method returns a LogRecordSequence with a length that is not equal to the parameter howMany.
3. Verify that the in/out parameter howMany is updated in the case where there are fewer records returned than the input parameter howMany. Verify this for the case where there are fewer than howMany records remaining in the log, beginning with currentID. (OE0700-C216, OE0700-C217)
   a. **Pass:** The returned LogRecordSequence will be of the length of the number of records returned. The in/out parameter howMany will contain the actual number of records returned in the LogRecordSequence. (must also pass step E.1.a and E.2.a)
   b. **Fail:** The *retrieve_records_by_producer_id* method returns a LogRecordSequence with a length that is not based upon the parameter howMany and/or the number of logs remaining.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)				
To be of assistance in locating the source code for the log service(s). Steps 1 thru 5 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the get_record_id_from_time operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.				
1. Locate and open the DMD file(s) for the OE.	DMD file opens.			
2. Determine the log service used, if any, recording the name of the service.	A log service is found.		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> Xxx </usesidentifier> <findby> <domainfinder type="**log**" name="**qqqqq**"> </domainfinder> </findby> </service> … </services>	
3. Locate and open the DCD file(s) for the OE.	DCD file opens.			

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate the log service from the DMD in the DCD file. Record the SPD file for the log service.	**Untested**: No log service is provided. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)		First, using the name from the DMD file, find in the DCD file: componentinstantiation refid using the following hierarchy: <componentplacement> <componentfileref refid="**rrrrr**"> </componentfileref> <componentinstantiation id="xxxx"> <usagename>**qqqqqq** </usagename> </componentinstantiation> </componentplacement> Then find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	
5. Identify any other log services provided in the DCD file and record the SPD files.	**Untested**: No log service is provided. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)		A log service can be identified in a DCD file by the following hierarchy: <connections>	

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
			… <connectinterface id="xxxxxx"> <usesport> <usesidentifier>LogPort </usesidentifier> <findby> <namingservice name="yyyyyy"> </namingservice> </findby> </usesport> <findby> <domainfinder type="**log**" name="**nnnn**"> </domainfinder> </findby> </connectinterface> … </connections> Find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
For each log service identified, perform the following steps.				
B. Verify that the log service provides the *retrieve_records_by_producer_id* method. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)				
6. Open the SPD file and determine the source code for the log service.	Source code is determined and located.		The name of the executable in the SCD file is the best clue for the name of the source code. Failure of these criteria OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, and OE0700-C219 means failure of the requirement OE0700.	
7. Verify that the log service provides the *retrieve_records_by_producer_id* method.	**Pass**: The *retrieve_records_by_producer_id* operation is present. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219) **Fail**: The *retrieve_records_by_producer_id* operation is not present. (OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, OE0700-C219)		Failure of these criteria OE0700-C214, OE0700-C215, OE0700-C216, OE0700-C217, OE0700-C218, and OE0700-C219 means failure of the requirement OE0700.	
C. Verify that the return value from *retrieve_records_by_producer_id* is of type LogRecordSequence and contains only records with a producer_id identical to the input/output currentId parameter**. (OE0700-C214)**				

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

8. Verify that the return value from *retrieve_records_by_producer_id* is of type LogRecordSequence.	**Pass**: The *retrieve_records_by_producer_id* method returns a LogRecordSequence. (OE0700-C214)
	
	Fail: The *retrieve_records_by_producer_id* method does not return a LogRecordSequence. (OE0700-C214)
			struct LogRecord {
			RecordId id;
			LogTime time;
			ProducerLogRecord info;
			};
			typedef
			sequence<LogRecord>
			LogRecordSequence;
			
			Failure of this criterion OE0700-C214 means failure of the requirement OE0700.	

9. Verify that the return value from *retrieve_records_by_producer_id* contains only records with a producer_id identical to the input/output currentId parameter**.**	**Pass**: The return value of *retrieve_records_by_producer_id* method contains only records with a producer_id identical to the input/output currentId parameter. (OE0700-C214)
	
	Fail: The return value of *retrieve_records_by_producer_id* method contains records with a producer_id not identical to the input/output currentId parameter. (OE0700-C214)
			struct LogRecord {
			RecordId id;
			LogTime time;
			ProducerLogRecord info;
			};
			typedef
			sequence<LogRecord>
			LogRecordSequence;
			
			Failure of this criterion OE0700-C214 means failure of the requirement OE0700.	

D. Verify the use of the currentId parameter. (OE0700-C215, OE0700-C218, OE0700-C219)				

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. Verify that the input parameter currentId is used by the *retrieve_records_by_producer_id* method to determine which records are returned.	**Pass**: The returned LogRecordSequence begins with the currentId. (OE0700-C215) **Fail**: The returned LogRecordSequence does not begin with the currentId. (OE0700-C215)		If the currentId corresponds to an existing log record, then that log record will be first in the returned LogRecordSequence. Failure of this criterion OE0700-C215 means failure of the requirement OE0700	
11. Test for the case where the input parameter currentId corresponds to the next record to be written (in the future). Verify that the returned LogRecordSequence is empty, and that the returned parameter howMany is zero, and that the returned parameter currentId is unchanged from the input value currentId.	**Pass**: The *retrieve_records_by_producer_id* method returns an empty LogRecordSequence, zero for howMany, or an unchanged currentId. (OE0700-C218) **Fail**: The *retrieve_records_by_producer_id* method does not return an empty LogRecordSequence, zero for howMany, or an unchanged currentId. (OE0700-C218)		Failure of this criterion OE0700-C218 means failure of the requirement OE0700	
12. Test for the case where the input parameter currentId corresponds to some record that is beyond the next record to be written. Verify that the returned LogRecordSequence is empty, and that the returned parameter howMany is zero, and that the returned parameter currentId is zero.	**Pass**: The *retrieve_records_by_producer_id* method returns an empty LogRecordSequence, zero for howMany, and zero for currentId. (OE0700-C219) **Fail**: The *retrieve_records_by_producer_id* method does not return an empty LogRecordSequence, zero for howMany, and zero for currentId. (OE0700-C219)		Failure of this criterion OE0700-C219 means failure of the requirement OE0700	

OE_TC_014				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
E. Verify the use of the howMany parameter. (OE0700-C216, OE0700-C217)				
13. Verify that the input parameter howMany is used by the *retrieve_records_by_producer_id* method to determine the number of records returned.	**Pass**: The LogRecordSequence returned indicates the length of howMany, (for the case where there are at least howMany records remaining) (OE0700-C216) **Fail**: The LogRecordSequence returned does not indicate the length of howMany. (for the case where there are at least howMany records remaining) (OE0700-C216)		This tests for the case where there are at least howMany records remaining in the log, beginning with currentID. Failure of this criterion OE0700-C216 means failure of the requirement OE0700	
14. Verify that the in/out parameter howMany is updated in the case where there are fewer records returned than the input parameter howMany.	**Pass**: The returned LogRecordSequence is the length of the number of records remaining in the log beginning with currentId, since there are fewer than howMany records remaining in the log. The in/out parameter howMany will contain the actual number of records returned in the LogRecordSequence. (OE0700-C216, OE0700-C217) **Fail**: The LogRecordSequence returned is not the length of the number of records remaining in the log. The in/out parameter howMany will contain the actual number of records returned in the LogRecordSequence. (OE0700-C216, OE0700-C217)		This tests for the case where there are fewer than howMany records remaining in the log, beginning with currentID. Failure of these criteria OE0700-C216 and OE0700-C217 means failure of the requirement OE0700	
End of Test				

Test Recording Log - OE_TC_014			
DMD File:			
Step 2 Log Service			
DCD File:			
Step 4 Log Service	Step 5 Log Service (DCD)		
SPD File:			
Step 7 Log Service			

Test Summary: OE_TC_014

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700_____

OE0700-C214 _____

OE0700-C215 _____

OE0700-C216 _____

OE0700-C217 _____

OE0700-C218_____

OE0700-C219_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.11. OE_TC_015 - OMG Lightweight Log Service :: retrieve_records_by_producer_name

Test Case Number: OE_TC_015

OMG Lightweight LOG Service::retrieve_records_by_producer_name

Requirements

SCA v2.2.2 Tag	SCA v2.2.2
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Specification
OE0700-C220	The *retrieve_records_by_producer_name* operation returns a LogRecordSequence of records that correspond to the supplied producerNames.
OE0700-C221	Candidate records for the LogRecordSequence begin with the record specified by the currentId parameter.
OE0700-C222	The number of records in the LogRecordSequence returned by the *retrieve_records_by_producer_name* operation is equal to the number of records specified by the howMany parameter, or the number of records available if the number of records specified by the howMany parameter cannot be met.
OE0700-C223	The log will update howMany to indicate the number of records returned and will set currentId to either the id of the record following the last examined record or the next record that will be recorded in the future if there are no further records available.
OE0700-C224	If the record specified by currentId does not exist, but corresponds to the next record that will be recorded in the future, the *retrieve_records_by_producer_name* operation returns an empty list of LogRecords, sets howMany to zero, and leaves the value of currentId unchanged.
OE0700-C225	If the record specified by currentId does not exist and does not correspond to the next record that will be recorded in the future, or if the Log is empty, the *retrieve_records_by_producer_name* operation returns an empty list of LogRecords, and sets both, currentId and howMany to zero.

References

Document Name	Version/Date	Location (Page, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005, Version 1.1, formal/05-02-02	Page 3-18, Section 3.3.2.5

Test Objective

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.

Places to Verify

Log Service

OMG Log Interfaces

Data

```
typedef unsigned long long RecordId;
typedef sequence<string> StringSeq;
struct LogRecord {RecordId id;
                  LogTime time;
                  ProducerLogRecord info; };
typedef sequence<LogRecord> LogRecordSequence;
```

Operations

```
LogRecordSequence retrieve_records_by_producer_name(
                                        inout RecordId currentId,
                                        inout unsigned long howMany,
                                        in StringSeq valueList);
```

Preconditions

- Domain profile files and all source code are available

Test Description

A. Determine whether log services are provided (OE0700, OE0700-C220, OE0700-C221, OE0700-C222, OE0700-C223, OE0700-C224, OE0700-C225)
   1. **Untested:** No log service is provided
B. For each log service identified in Step A, perform the following steps:
   1. Validate that the *retrieve_records_by_producer_name* operation returns the LogRecordSequence of records corresponding to the supplied producerNames. Note that it is a set of producerNames (OE0700-C220)
      a. **Pass:** Returns a LogRecordSequence of records that corresponds to the producerNames
      b. **Fail:** A LogRecordSequence is not returned
      c. **Fail:** One or more records are returned with an invalid producerName
   2. Verify that the RecordId's in the LogRecordSequence start with or after the currentId parameter (OE0700-C221)

   a. **Pass:** The RecordId's are valid
   b. **Fail:** The RecordId's are not valid
3. Verify that the number of records in the LogRecordSequence is equal to the input howMany parameter, or less than it if there are not that many valid records available (OE0700-C222)
   a. **Pass:** The number of records is valid
   b. **Fail:** The number of records is greater than the input howMany parameter or is not equal to the number of valid records available
4. Verify that the howMany parameter indicates the number of RecordId's in the LogRecordSequence when the operation returns (OE0700-C223)
   a. **Pass:** Number of RecordId's in LogRecordSequence and the returned howMany parameter agree
   b. **Fail:** Number of RecordId's in LogRecordSequence and the returned howMany parameter do not agree
5. Verify that the currentId points to the next record after the last record in the LogRecordSequence or to the next record to be written if the number of valid records was less than the input value of the howMany parameter (OE0700-C223)
   a. **Pass:** The returned value of currentId points to the record after the last record in LogRecordSequence or The returned value of the currentId points to the next record to be written
   b. **Fail:** The returned value of the currentId does not point to the next record to be written or the record after the last RecordId in LogRecordSequence
6. Verify that the value of currentId is unchanged and the LogRecordSequence is empty if the record specified by input currentId does not exist, but corresponds to the next record that will be recorded (OE0700-C224)
   a. **Pass:** The returned value of currentId is the same as the input value, howMany is zero, and the LogRecordSequence is empty
   b. **Fail:** The returned value of RecordId is not the same as its input value
   c. **Fail:** The LogRecordSequence is not empty
7. Verify that the return value of currentId is 0, howMany is 0 and the LogRecordSequence is empty if the record specified by input currentId does not exist and the value of currentId does not correspond to the next record to be written (OE0700-C225)
   a. **Pass:** The return value of currentId is zero, howMany is zero and the LogRecordSequence is empty
   b. **Fail:** The return value of currentId is not 0
   c. **Fail:** The return value of howMany is not 0
   d. **Fail:** The LogRecordSequence is not empty

Manual Test Steps

Notes: 1. Test Result will include PASS, FAIL, Untested, or N/A.
2. The Test Recording Log is intended to record long comments and lists of SPD files, SCD files, PRF, and other file/data.

NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). Steps 1 thru 5 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the retrieve_records_by_producer_name operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700, OE0700-C220, OE0700-C221, OE0700-C222, OE0700-C223, OE0700-C224, OE0700-C225)				
1. Locate and open the DMD file(s) for the OE.	DMD file opens.			
2. Determine the log service used, if any, recording the name of the service.	A log service is found. **Untested:** No log service is provided. (OE0700, OE0700-C220, OE0700-C221, OE0700-C222, OE0700-C223, OE0700-C224, OE0700-C225)		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> Xxx </usesidentifier> <findby> <domainfindertype="**log**" name="**qqqqqq**"> </domainfinder> </findby> </service> … </services>	
3. Locate and open the DCD file(s) for the OE.	DCD file opens.			

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate the log service from the DMD in the DCD file. Record the SPD file for the log service.	The log service is found.		First, using the name from the DMD file, find in the DCD file: componentinstantiation refid using the following hierarchy: <componentplacement> <componentfileref refid="**rrrrr**"> </componentfileref> <componentinstantiation id="xxxx"> <usagename>**qqqqqq** </usagename> </componentinstantiation> </componentplacement> Then find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
5. Identify any other log services provided in the DCD file.	All other log services are located.		A log service can be identified in a DCD file: <connections> … <connectinterface id="xxxxxx"> <namingservice name="yyyyyy"> </namingservice> <domainfinder type="**log**" name="**nnnn**"> </domainfinder> </findby> </connectinterface> </connections>	
6. Record the SPD file in the DCD file.			Find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	
For each log service identified, perform the following steps.				

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
B.1. Validate that the *retrieve_records_by_producer_name* operation returns the LogRecordSequence of records corresponding to the supplied producerNames. Note that it is a set of producerNames. (OE0700-C220)				
7. Determine the source code for the log service.	**Pass:** The source code is found. (OE0700-C220) **Fail:** The source code is not found. (OE0700-C220)		Source code is determined and located. Failure of this criterion OE0700-C220 means failure of the requirement OE0700.	
8. Identify the source code for retrieve_*records_by_producer_name* operation.	**Pass:** The source code for *retrieve_records_by_producer_name* operation is found. (OE0700-C220) **Fail:** The source code for *retrieve_records_by_producer_name* operation is not found. (OE0700-C220)		Failure of this criterion OE0700-C220 means failure of the requirement OE0700.	
9. Verify that the source code returns the LogRecordSequence of records.	**Pass:** returns a LogRecordSequence of records that corresponds to the producerNames. (OE0700-C220) **Fail:** A LogRecordSequence is not returned. (OE0700-C220) **Fail:** One or more records are retumed with an invalid ProducerName. (OE0700-C220)		Failure of this criterion OE0700-C220 means failure of the requirement OE0700.	
B.2. Verify that the RecordId's in the LogRecordSequence start with or after the currentId parameter. (OE0700-C221)				

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. Verify the currentID.	**Pass:** The RecordId's are valid. (OE0700-C221) **Fail:** The RecordId's are not valid. (OE0700-C221) Write it down in the Log Sheet.		As there is no ordering requirement for RecordId, "after" refers to the logTime of the record Failure of this criterion OE0700-C221 means failure of the requirement OE0700.	
11. Verify that the RecordId's in the LogRecordSequence start with or after the currentId parameter.	**Pass:** The RecordId starts at or after currentID. (OE0700-C221) **Fail:** The RecordId does not start at or after currentID. (OE0700-C221)		Failure of this criterion OE0700-C221 means failure of the requirement OE0700.	
B.3. Verify that the number of records in the LogRecordSequence is equal to the input howMany parameter, or less than it if there are not that many valid records available. (OE0700-C222)				
12. Verify that the proper number of RecordId's are returned.	**Pass:** The number of records is valid (OE0700-C222) **Fail:** The number of records is greater than the input howMany parameter or is not equal to the number of valid records available. (OE0700-C222)		Failure of this criterion OE0700-C222 means failure of the requirement OE0700.	
B.4. Verify that the howMany parameter indicates the number of RecordId's in the LogRecordSequence when the operation returns. (OE0700-C223)				
13. Verify that the howMany parameter indicates the number of RecordId's in the LogRecordSequence when the operation returns.	**Pass:** Number of RecordId's in LogRecordSequence and the returned howMany parameter agree. (OE0700-C223) **Fail:** Number of RecordId's in LogRecordSequence and the returned howMany parameter do not agree. (OE0700-C223)		Failure of this criterion OE0700-C223 means failure of the requirement OE0700.	

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
B.5. Verify that the currentId points to the next record after the last record in the LogRecordSequence or to the next record to be written if the number of valid records was less than the input value of the howMany parameter. (OE0700-C223)				
14. Verify that the proper currentId is returned.	**Pass:** The returned value of currentId points to the record after the last record in LogRecordSequence or the returned value of the currentId points to the next record to be written. (OE0700-C223) **Fail:** The returned value of the currentId does not point to the next record to be written or the record after the last RecordId in LogRecordSequence. (OE0700-C223)		Failure of this criterion OE0700-C223 means failure of the requirement OE0700.	
B.6. Verify that the value of currentId is unchanged and the LogRecordSequence is empty if the record specified by input currentId does not exist, but corresponds to the next record that will be recorded. (OE0700-C224)				
15. Verify that the proper currentId value and LogRecordSequence value are returned when the currentId record is the next record to be written.	**Pass:** The returned value of currentId is the same as the input value, howMany is zero, and the LogRecordSequence is empty. (OE0700-C224) **Fail:** The returned value of RecordId is not the same as its input value. (OE0700-C224) **Fail:** The LogRecordSequence is not empty (OE0700-C224)		Failure of this criterion OE0700-C224 means failure of the requirement OE0700.	
16. Verify that the howMany parameter is zero.	**Pass: The** howMany is set to 0. (OE0700-C224) **Fail: The** howMany is not set to 0. (OE0700-C224)		Failure of this criterion OE0700-C224 means failure of the requirement OE0700.	
B.7. Verify that the return value of currentId is 0, howMany is 0 and the LogRecordSequence is empty if the record specified by input currentId does not exist and the value of currentId does not correspond to the next record to be written. (OE0700-C225)				

OE_TC_015				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
17. Verify that the proper currentId value and LogRecordSequence value are returned when the currentId record does not exist and is not the next record to be written.	**Pass:** The return value of currentId is zero, howMany is zero and the LogRecordSequence is empty. (OE0700-C225) **Fail:** The return value of currentId is not 0. (OE0700-C225)		Failure of this criterion OE0700-C225 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_015					
DMD File					
Step 2 (log service in DMD)	**Step3 (DCD file)**	**Step4 (log service in DCD)**	**Step5 (log services)**	**Step6 (SPD file)**	**Step7 (log service)**
Step 8 (*retrieve_records_by_producer_name* operation)	Step 9 (LogRecordSequence)	Step 10 (currentId)	Step 11 (after currentId)	Step 12 (number of RecordId's returned)	

Test Summary: OE_TC_015

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C220 _____

OE0700-C221 _____

OE0700-C222 _____

OE0700-C223 _____

OE0700-C224 _____

OE0700-C225 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.12. OE_TC_016 - OMG Lightweight Log Service :: write_records

Test Case Number: OE_TC_016

OMG Lightweight Log Service::LogProducer::LogStatus::write_records

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification (OE700C36 – OE700C40 list below).
	OMG Lightweight Log Specification
OE0700-C226	The *write_records* operation adds the log records supplied in the records parameter to the Log.
OE0700-C227	When there is insufficient storage to add one of the supplied log records to the Log, and the *LogFullAction* is set to HALT, the *write_records* operation will set the availability status logFull state to TRUE.
OE0700-C240	For example, if 3 records are provided in the records parameter, and while trying to write the second record to the log, the record will not fit, then the log is considered to be full. Therefore, the second and third records will not be stored in the log but the first record would have been successfully stored.
OE0700-C228	When there is insufficient storage to add one of the supplied log records to the Log, and the *LogFullAction* is set to WRAP, the *write_records* operation will overwrite the oldest *LogRecords* with the newest records, as they are written to the Log, and leave the availability status *logFull* state unchanged.
OE0700-C229	The *write_records* operation inserts the current UTC time to the time field of each record written to the Log, and assigns a unique record id to the id field of the LogRecord.
OE0700-C230	Log records accepted for storage by the *write_records* will be available for retrieval in the order received.

References

Document Name	Version/Date	Location (Pages, Section)
SCA	Version 2.2.2 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Pages 3-20 – 3-21, Section 3.3.3.1

Test Objective

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. 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.

Places to Verify

Core Framework Log Service

OMG Log Interfaces

Data

```
struct ProducerLogRecord {    string producerId;
                              string producerName;
                              LogLevel level;
                              string logData; };
typedef sequence<ProducerLogRecord> ProducerLogRecordSequence;
```

Operations

void write_records(in ProducerLogRecordSequence records);

Preconditions

- All the Domain profile files are available
- The source code files are available.

Test Description

A. Determine from the domain profile whether log services are provided.(OE0700, OE0700-C226, OE0700-C227, OE0700-C228, OE0700-C229, OE0700-C230, OE0700-C240)
   1. **Untested**: No log service is provided.

B. Verify that the *write_records()* operation adds the Log records supplied in its parameter to the Log. (OE0700-C226)
   1. **Pass:** The *write_records()* adds the Log records in the parameter to the Logging Storage Area.
   2. **Fail:** The *write_records()* operation does *not* add the Log records supplied in its parameter to the logging storage area.

C. Verify that the *write_records()* operation sets the availability status, *logFull*, to TRUE when there is not enough storage to write the log records from the parameter and the *LogFullAction* is set to HALT. Verify that the operation does not write the records for which there is insufficient space. The log records in the parameter can be one or more in number. (OE0700-C227, OE0700-C240)
   1. Verify that the *write_records()* operation sets the availability status, *logFull*, to true when *LogFullAction* is set to HALT and there is insufficient space to hold the records. (OE0700-C227)

a. **Pass:** When there is insufficient storage in the *LogStorageArea* and the *LogFullAction* is set to HALT, the *write_records()* operation sets the *LogFull* state to TRUE.
b. **Fail**: When there is insufficient storage in the *LogStorageArea* and the *LogFullAction* is set to HALT, the *write_records()* operation does not set the *logFull* state to TRUE.

2. Verify that the *write_records()* operation is not allowed to write records to the Log when there is not enough space for these records and *LogFullAction* is set to HALT. This would uphold the following statement from the Lightweight Log: "For example, if 3 records are provided in the records parameter, and while trying to write the second record to the log, the record will not fit, then the log is considered to be full. Therefore, the second and third records will not be stored in the log but the first record would have been successfully stored". (OE0700-C240)
   a. **Pass:** The *write_records()* operation only writes the records for which there is sufficient space and does not overwrite existing records in the Log.
   b. **Fail:** The *write_records()* operation is allowed to write additional records to the Log when there is not enough space for these records.

D. Verify that the *write_records()* operation overwrites the oldest records in the order that they were received and does not change the availability status, *logFull*, when the *LogFullAction* is set to WRAP and there is insufficient storage in the Log. (OE0700-C228)
NOTE: In order to fully test the *write_records()* operation when *LogFullAction* is set to WRAP, boundary conditions should be considered. The boundary conditions could include the cases when the *write_records()* operation has written to the available space in the log and the remaining records in the parameter are less than, equal to or more than the current number of records contained in the Log. For all situations, the *write_records()* operation should uphold the requirement even though the Lightweight Log specification does not state specifically how the operation should behave in various boundary cases.
   1. **Pass:** When the *LogFullAction* is set to WRAP and there is insufficient storage in the Log, the *write_records()* operation successfully overwrites the oldest *LogRecords* in the log with the newest records from the operation's parameter, overwriting the records in the order they were received.
   2. **Fail:** The *write_records()* operation does *not* overwrite the older records in the log with the newest records in the order they were received
   3. **Fail**: The *write_records()* operation changes the availability status, *logFull*, when there is insufficient storage in the Log and the *LogFullAction* is set to WRAP.

E. Verify that the *write_records()* operation inserts the current UTC time to the time field of each record written to the Log, and assigns a unique record id to the id field of the LogRecord. (OE0700-C229)
   1. **Pass:** The *write_records()* operation inserts the current UTC time in the time field of each record written to the log and assigns a unique record id in the id field of the LogRecord.
   2. **Fail:** The *write_records*() operation does *not* insert the current UTC time in the time field of each record written to the log.
   3. **Fail:** The *write_records()* operation does *not* assign a unique record id in the id field of the LogRecord.

F. Verify that the Log records accepted for storage from the *write_records()* operation will be made available for retrieval in the order they are received in the Log. (OE0700-C230)
   1. **Pass:** The records, accepted for storage from the *write_records()* operation, are made available for retrieval in the order they are received.
   2. **Fail:** The records, accepted for storage by the *write_records()* operation, are *not* made available for retrieval in the order they are received.

Manual Test Steps

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

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

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the Domain Profile whether Log Services are provided. (OE0700, OE0700-C226, OE0700-C227, OE0700-C228, OE0700-C229, OE0700-C230, OE0700-C240)				
NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). If the OE software engineer is not available, then steps 1 thru 7 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.				
1. Locate and open the DMD file(s) for the OE.	DMD file opens. Note: The DMD file may be named DomainManager.dmd.xml			
2. Determine the log service(s) used, if any, recording the name of the service.	Log service(s) found. **Untested**: No log service found. (OE0700, OE0700-C226, OE0700-C227, OE0700-C228, OE0700-C229, OE0700-C230, OE0700-C240) Note: Most likely, the name of the log service is LogService. In our example, we are using qqqqq.		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> xxxxx </usesidentifier> <findby> <domainfinder type= "log" name="qqqqq"> </domainfinder> </findby> </service> … </services>	
3. Locate and open the DCD file(s) for the OE.	DCD file opens.		The Domain DCD file may be named DeviceManager.dcd.xml.	

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate log service *componentplacement* declaration in the DCD file using the name of the log service from the DMD and record the *componentfileref refid*.	*componentplacement* declaration for declared log service exists in the DCD file.		The name of the log service will be a usagename in the componentplacement section of the DCD. XML declaration example: <componentplacement> <componentfileref refid="**qqqqq_File**"> </componentfileref> <componentinstantiation id="**xxxxx**" <usagename>**qqqqq** </usagename> </componentinstantiation> </componentplacement>	
5. Locate the log service *componentfile id* declaration *using the componentfileref refid* and record the related SPD file name.	*Componentfile id* declaration for declared log service exists in the DCD file.		The refid from step 4 will be the id in the componentfile section of the DCD. XML declaration example: <componentfiles> <componentfile id=**"qqqqq_File"** type="Software Package Descriptor"> <**localfile name="qqqqq.spd.xml"/>** </componentfile>	
6. Examine the SPD files (step 5) and record the localfile name.	The log service executable declaration exists.		XML declaration example: <implementation id="xxxx"> <code type= "xxxxx"> <localfile name= "…/qqqqq.exe"/>	

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
B. Verify that the *write_records()* operation adds the Log records supplied in its parameter to the Log. (OE0700-C226)				
7. Examine the source code for the log service and verify that *write_records()* operation exists.	**Pass**: The *write_records()* operation exists. (OE0700-C226) **Fail**: The *write_records()* operation does not exist. (OE0700-C226)		Failure of this criterion OE0700-C226 means failure of the requirement OE0700.	
8. Verify in the source code that the *write_records()* operation adds the Log records supplied in its parameter to the logging storage area.	**Pass:** The *write_records()* operation must add the Log records supplied its parameter to the logging storage area. (OE0700-C226) **Fail:** The *write_records()* operation does not add the Log records supplied its parameter to the logging storage area. (OE0700-C226)		Failure of this criterion OE0700-C226 means failure of the requirement OE0700.	
C. Verify that the write_records() operation sets the availability status, logFull, to TRUE when there is not enough storage to write the log records from the parameter and the LogFullAction is set to HALT. Verify that the operation does not write the records for which there is insufficient space. The log records in the parameter can be one or more in number. (OE0700-C227, OE0700-C240) **1. Verify that the *write_records()* operation sets the availability status, *logFull*, to true when *LogFullAction* is set to HALT and there is insufficient space to hold the records.** **Verify that the *write_records()* operation is not allowed to write records to the Log when there is not enough space for these records and *LogFullAction* is set to HALT. This is upholding the following statement from the Lightweight Log: "For example, if 3 records are provided in the records parameter, and while trying to write the second record to the log, the record will not fit, then the log is considered to be full. Therefore, the second and third records will not be stored in the log but the first record.**				

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
9. Verify in the source code that the *write_records()* operation sets the availability status, *logFull*, to TRUE when there is not enough storage to add the next supplied log record and *LogFullAction* is set to HALT	**Pass**: The *write_records()* operation sets the availability status, *logFull*, to TRUE. (OE0700-C227) **Fail**: The write_records() operation does not set the availability status, logFull, to TRUE. (OE0700-C227)		One possible case to consider is the additional description from the OMG Lightweight Log on page 3-20 "For example, if 3 records are provided in the records parameter, and while trying to write the second record to the log, the record will not fit, then the log is considered full. Therefore, the second and third records will not be stored in the log but the first record would have been successfully stored." Failure of this criterion OE0700-C227 means failure of the requirement OE0700.	

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. When the *LogFullAction* is set to HALT, verify in the source code that the *write_records()* operation is prevented from writing records to the Log for which there is insufficient space. *The write_records()* operation should perform correctly under the following scenario: "For example, if 3 records are provided in the records parameter, and while trying to write the second record to the log, the record will not fit, then the log is considered to be full. Therefore, the second and third records will not be stored in the log but the first record would have been successfully stored".	**Pass**: The *write_records()* operation is prevented from writing the records to the Log when there is insufficient space. (OE0700-C240) **Fail**: The *write_records()* operation is not prevented from writing the records to the Log when there is insufficient space. (OE0700-C240)		Failure of this criterion OE0700-C240 means failure of the requirement OE0700.	
D. Verify that the *write_records()* operation overwrites the oldest records in the order that they were received and does not change the availability status, *logFull*, when the *LogFullAction* is set to WRAP and there is insufficient storage in the Log. (OE0700-C228) **NOTE: In order to fully test the *write_records()* operation when *LogFullAction* is set to WRAP, boundary conditions should be considered. The boundary conditions could include the cases when the *write_records()* operation has written to the available space in the log and the remaining records in the parameter are less than, equal to or more than the current number of records contained in the Log. For all possible situations, the *write_records()* operation should uphold the requirement even though the Lightweight Log specification does not state specifically how the operation should behave in various boundary cases.**				

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
11. Verify that the *write_records()* operation overwrites the oldest record when the log is full and the *LogFullAction* is set to WRAP. The next oldest record then becomes the oldest record. **NOTE**: Possible cases to consider are when the *write_records()* operation has written to the available space in the Log but its remaining records from the parameter are less than, equal to or more than the number of records in the Log.	**Pass**: The *write_records()* operation overwrites the oldest log record(s) and replaces them with the records in the operation's parameter. (OE0700-C228) **Fail**: The write_records() operation does not overwrite the oldest log record(s) and replaces them with the records in the operation's parameter. (OE0700-C228)		A circular queue is the most likely implementation for WRAP. If the implementation uses a circular queue then the oldest record is located at the "tail" pointer and the tail pointer would move when the log is filled and the LogFullAction is set to WRAP. In a WRAP condition, the log remains full after the oldest record has been overwritten. Failure of this criterion OE0700-C228 means failure of the requirement OE0700.	
12. Verify that the availability status, logFull, is not changed by the *write_records()* operation when the log is full and the *LogFullAction* is set to wrap. **NOTE**: Cases to consider are when *the logFull* is initially set to either TRUE or FALSE.	**Pass**: The *write_records()* operation does not change the availability status, *logFull*, when the log is full and the *LogFullAction* is set to wrap. (OE0700-C228) **Fail:** The *write_records()* operation changes the availability status, logFull, when the log is full and the LogFullAction is set to wrap. (OE0700-C228)		Failure of this criterion OE0700-C228 means failure of the requirement OE0700.	
E. Verify that the write_records() operation inserts the current UTC time to the time field of each record written to the Log, and assigns a unique record id to the id field of the LogRecord. (OE0700-C229)				

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
13. Verify in the source code that the *write_records()* operation inserts the current UTC time to the time field of each record written to the Log.	**Pass**: The *write_records()* operation inserts the current UTC time to the time field of each record written to the Log. (OE0700-C229) **Fail**: The *write_records()* operation does not insert the current UTC time to the time field of each record written to the Log. (OE0700-C229)		The current UTC time is most likely retrieved from the OE timing service. Failure of this criterion OE0700-C229 means failure of the requirement OE0700.	
14. Verify in the source code that the *write_records()* operation assigns a unique record id to the id field of the LogRecord for each record written in the Log.	**Pass**: The *write_records()* operation assigns a unique record id to the id field of the LogRecord for each record written in the Log. (OE0700-C229) **Fail:** The *write_records()* operation does not assign a unique record id to the field of the LogRecord for each record written in the Log. (OE0700-C229)		The Log specification defines RecordId as a 64-bit type (unsigned long long in C). All computations of RecordId should involve operations with compatible types. Uniqueness is reasonably accomplished by an incrementing counter, but record ids are not specifically required to be sequential nor increasing. Failure of this criterion OE0700-C229 means failure of the requirement OE0700.	
F. Verify that the Log records accepted for storage by the *write_records()* operation will be made available for retrieval in the order they are received. (OE0700-C230)				

OE_TC_016				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
15. Verify in the source code that the Log records accepted for storage from the *write_records()* operation will be made available for retrieval in the order they are received.	**Pass**: The Log records, accepted for storage from the *write_records()* operation, are made available for retrieval in the order they are received. (OE0700-C230) **Fail**: The Log records, accepted for storage from the write_records() operation, are not made available for retrieval in the order they are received. (OE0700-C230)		Failure of this criterion OE0700-C230 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_016			
DCD file:			
SPD file:			
Step 1-3. (List Log Service(s))	Step 4 (SPD file names)	Step 5 & 6 (Executable Source File Names)	*Steps 7 & 8* *(Write_records()* adds *LogRecords* to Log)
Step 9 & 10 (Write_records operation sets LogFull to TRUE when insufficient storage is in the Log and LogFull is set to HALT)	Steps 11 & 12 (Write_records() over-writes oldest logs and leaves LogFull unchanged when the logFull is set to WRAP)	Steps 13 & 14 (Write_records() sets UTC time and unique record id)	Steps 15 (Logs set for storage by the write_records are made retrievable in the order they are received)

Test Summary: OE_TC_016

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s)(x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C226 _____

OE0700-C227 _____

OE0700-C228 _____

OE0700-C229 _____

OE0700-C230 _____

OE0700-C240 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.13. OE_TC_017 - OMG Lightweight Log Service :: write_record

Test Case Number: OE_TC_017

OMG Lightweight Log Service::write_record

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification.
	OMG Lightweight Log Specification
OE0700-C231	The *write_record* operation adds a log record supplied in the record parameter to the Log.
OE0700-C232	When there is insufficient storage to add the supplied log record to the Log, and the LogFullAction is set to HALT, the *write_record* operation will set the availability status logFull state to true.
OE0700-C233	When there is insufficient storage to add the supplied log record to the Log, and the LogFullAction is set to WRAP, the *write_record* operation will overwrite the oldest LogRecords with the new record, and leave the availability status logFull state unchanged.
OE0700-C234	The *write_record* operation inserts the current UTC time to the time field of each record written to the Log, and assigns a unique record id to the id field of the LogRecord.
OE0700-C235	Log records accepted for storage by *write_record* will be available for retrieval in the order received.

References

Document Name	Version/Date	Pages referenced
Software Communications Architecture (SCA)	Version 2.2.2 / 05-15-2006	Page 3-2, Section 3.1.2.2
OMG Lightweight Log Service Specification	Version 1.1 / 02-02-2005	Pages 2-19 and 3-21 to 3-22

Test Objective

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. 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).

Places to Verify

Core Framework Log Service

OMG Log Interfaces

Data

```
typedef unsigned short LogLevel;
struct ProducerLogRecord {    string producerId;
                              string producerName;
                              LogLevel level;
                              string logData; };
```

Operations

```
void write_record(in ProducerLogRecord record);
```

Preconditions

- All of the domain profile files and Operating Environment (OE) source code files are available.

Test Description

A. Determine whether log services are provided. (OE0700, OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)
   1. **Untested:** No log service is provided.

For each log service identified in Step A, perform the following steps:

B. Verify that the *write_record* operation is present. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)
   1. **Pass:** The *write_record* operation is present
   2. **Fail:** The *write_record* operation is not present.

C. Verify that the *write_record* operation tests whether the log is full or not full before adding the additional log entry. (OE0700-C232, OE0700-C233)
   1. **Pass:** The *write_record* operation checks for the condition of the log being full.
   2. **Fail:** The *write_record* operation does not check for the condition of the log being full.

D. Verify that when there is insufficient storage to add the supplied log record to the Log, and the LogFullAction is set to HALT, then the *write_record* operation sets the availability status logFull state to true and does not write the record. (OE0700-C232)
   1. **Pass:** The *write_record* operation sets the availability status logFull state to true and does not write the record.
   2. **Fail:** The *write_record* operation does not set the logFull state to true.
   3. **Fail:** The *write_record* operation writes the record to the log.

E. Verify that the *write_record* operation overwrites the oldest record when the log is full and the LogFullAction is set to WRAP. (OE0700-C233)
   1. **Pass:** The *write_record* operation overwrites the oldest record when the log is full and the LogFullAction is set to WRAP.
   2. **Fail:** The *write_record* operation does not write the record when the log is full and the LogFullAction is set to WRAP.
   3. **Fail:** The *write_record* operation overwrites any record other than the oldest when the log is full and the LogFullAction is set to WRAP.

F. Verify that the *write_record* operation writes the record when the log is not full. (OE0700-C231)
   1. **Pass:** The *write_record* operation writes the record when the log is not full.
   2. **Fail:** The *write_record* operation does not write the record (it does not get stored).
   3. **Fail:** The *write_record* operation writes the record to an existing record's location (overwrites an existing log record) when the log is not full.

G. Verify that the *write_record* operation obtains the current time (UTC) from the system and writes that into the time field of the log record. (OE0700-C234)
   1. **Pass:** The *write_record* operation obtains the current time (UTC) from the system and writes that into the time field of the log record.
   2. **Fail:** The *write_record* operation does not obtain the current time (UTC) from the system or does not write that into the time field of the log record.

H. Verify that the *write_record* operation assigns a unique record id to the id field of the log record. (OE0700-C234)
   1. **Pass:** The *write_record* operation assigns a unique record id to the id field of the log record.
   2. **Fail:** The *write_record* operation does not assign a log record id or reuses a previous log record id.

I. Verify that the *write_record* operation makes the records available for retrieval in the order received. The challenging test case to see whether this is implemented properly is to consider what happens when the log is filled and the LogFullAction is set to WRAP. (OE0700-C235)
   1. **Pass:** The *write_record* operation keeps track of the oldest log record and it is updated if the log is filled and the LogFullAction is set to WRAP.
   2. **Fail:** There are any circumstances (e.g. log full) where *write_record* operation will place records in an order that will not result in the records being retrieved in order, beginning with the oldest record.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log sheet is intended to record data for each step that requires recording of data.

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700, OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)				
NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). Steps 1 thru 5 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the get_record_id_from_time operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.				
1. Locate and open the DMD file(s) for the OE.	DMD file opens.			
2. Determine the log service used, if any, recording the name of the service.	A log service is found. **Untested**: No log service is found. (OE0700, OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> Xxx </usesidentifier> <findby> <domainfindertype="**log**" name="**qqqqqq**"> </domainfinder> </findby> </service> … </services>	
3. Locate and open the DCD file(s) for the OE.	DCD file opens.			

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate the log service from the DMD in the DCD file. Record the SPD file for the log service.	The log service is found. **Untested**: No log service is found. (OE0700, OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)		First, using the name from the DMD file, find in the DCD file: componentinstantiation refid using the following hierarchy: <componentplacement> <componentfileref refid="**rrrrr**"> </componentfileref> <componentinstantiation id="xxxx"> <usagename>**qqqqqq** </usagename> </componentinstantiation> </componentplacement> Then find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	
5. Identify any other log services provided in the DCD file and record the SPD files.	All other log services are located.		A log service identified in a DCD file by the following hierarchy: <connections> … <connectinterface id="xxxxxx"> <usesport> <usesidentifier>LogPort	

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
			</usesidentifier> <findby> <namingservice name="yyyyyy"> </namingservice> </findby> </usesport> <findby> <domainfinder type="**log**" name="**nnnn**"> </domainfinder> </findby> </connectinterface> … </connections> Find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> /componentfiles>	
For each log service identified, perform the following steps.				
B. Verify that the *write_record* operation is present. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)				

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Open the SPD file and determine the source code location for the log service. Open the log service source code.	**Pass**: Source code is located. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235) **Fail**: Source code is not located. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)		The name of the executable in the SPD file is the best clue for the name of the source code. Failure of these criteria OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235 means failure of the requirement OE0700.	
7. Verify that the *write_record* operation is present in the source code.	**Pass**: The *write_record* operation is present. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235) **Fail**: The *write_record* operation is not present. (OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235)		Failure of these criteria OE0700-C231, OE0700-C232, OE0700-C233, OE0700-C234, and OE0700-C235 means failure of the requirement OE0700.	
C. Verify that the *write_record* operation tests whether the log is full or not full before adding the additional log entry. (OE0700-C232, OE0700-C233)				
8. Examine the *write_record* code to see how it checks for a full log.			If necessary, this may include looking at the entire LogService source code for context to determine how log records are stored and how the log service maintains a count. Since it is possible to have the log "WRAP" when full, a circular queue is a likely implementation.	

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
9. Verify that the *write_record* operation tests whether the log is full or not full before adding the additional log entry.	**Pass**: The *write_record* operation checks for the condition of the log being full. (OE0700-C232 and OE0700-C233) **Fail**: The *write_record* operation checks for the condition of the log being full. (OE0700-C232 and OE0700-C233)		Failure of these criteria OE0700-C232 and OE0700-C233 means failure of the requirement OE0700.	
D. Verify that when there is insufficient storage to add the supplied log record to the Log, and the LogFullAction is set to HALT, then the *write_record* operation sets the availability status logFull state to true and does not write the record. (OE0700-C232)				
10. Examine the Log Service code to see how the LogFullAction is set and how this value is used during the *write_record* operation.				
11. Verify that when there is insufficient storage to add the supplied log record to the Log, and the LogFullAction is set to HALT, then the *write_record* operation sets the availability status logFull state to true and does not write the record.	**Pass**: The *write_record* operation sets the logFull state to true. The *write_record* operation does not write the record when the log is full and the LogFullAction is set to HALT. (OE0700-C232) **Fail**: The *write_record* operation sets the logFull state to true. (OE0700-C232) **Fail**: The *write_record* operation writes the record to the log is full. (OE0700-C232)		Failure of this criterion OE0700-C232 means failure of the requirement OE0700.	
E. Verify that the *write_record* operation overwrites the oldest record when the log is full and the LogFullAction is set to WRAP. (OE0700-C233)				

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
12. Examine the Log Service implementation to determine where the oldest record would be in a log.			If the implementation uses a circular queue then the oldest record is located at the "tail" pointer and the tail pointer would move when the log is filled and the LogFullAction is set to WRAP. In a WRAP condition, the log remains full after the oldest record has been overwritten.	
13. Verify that *the write_record* operation overwrites the oldest record when the log is full and the *LogFullAction* is set to WRAP. The next oldest record then becomes the oldest record.	**Pass**: The *write_record* operation overwrites the oldest log record when the log is filled and the *LogFullAction* is set to WRAP. (OE0700-C233) **Fail**: The *write_record* operation does not overwrite the oldest log record when the log is filled and the *LogFullAction* is set to WRAP. (OE0700-C233) **Fail**: The *write_record* operation overwrites any record other than the oldest log record when the log is filled and the *LogFullAction* is set to WRAP. (OE0700-C233)		Failure of this criterion OE0700-C233 means failure of the requirement OE0700.	
F. Verify that the *write_record* operation writes the record when the log is not full. (OE0700-C231)				

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
14. Examine the implementation for the typical case where the log is not full. Verify that the *write_record* operation writes the record when the log is not full.	**Pass**: The *write_record* operation writes the record when the log is not full. (OE0700-C231) **Fail**: The *write_record* operation does not write the record when the log is not full. (OE0700-C231) **Fail**: The *write_record* operation writes the record to an existing record's location when the log is not full. (OE0700-C231)		Failure of this criterion OE0700-C231 means failure of the requirement OE0700.	
G. Verify that the *write_record* operation obtains the current time (UTC) from the system and writes that into the time field of the log record. (OE0700-C234)				
15. Examine the OE timing service from which the current time (UTC) is obtained.				
16. Verify that the *write_record* operation obtains the current time (UTC) from the system and writes that into the time field of the log record.	**Pass**: The *write_record* operation obtains the current time (UTC) from the system and writes that into the time field of the log record. (OE0700-C234) **Fail**: The *write_record* operation does not obtain the current time (UTC) from the system and writes that into the time field of the log record. (OE0700-C234)		Failure of this criterion OE0700-C234 means failure of the requirement OE0700.	
H. Verify that the *write_record* operation assigns a unique record id to the id field of the log record. (OE0700-C234)				

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
17. Examine the implementation of the Log Service to see how it handles the record id.			The Log specification gives the RecordId type as a 64-bit (unsigned long long in C). All computations of RecordId should involve operations with compatible types. Uniqueness is reasonably accomplished by an incrementing counter, but record ids are required to be neither sequential nor increasing.	
18. Verify that the *write_record* operation assigns a unique record id the id field of the log record.	**Pass**: The *write_record* operation assigns a unique record id to the id field of the log record. (OE0700-C234) **Fail**: The *write_record* operation does not assign a unique record id to the id field of the log record. (OE0700-C234)		Failure of this criterion OE0700-C234 means failure of the requirement OE0700.	
I. Verify that the *write_record* operation makes the records available for retrieval in the order received. The challenging test case to see whether this is implemented properly is to consider what happens when the log is filled and the LogFullAction is set to WRAP. (OE0700-C235)				
19. Examine the implementation to determine how the oldest record is tracked (e.g. by the "tail" pointer in a circular queue). The record retrieval operations must be able to retrieve records in the order received.	The *write_record* operation tracks the oldest log record and this is updated when a WRAP occurs.		The challenging test case for whether this is implemented properly is to consider what happens when the log is filled and the LogFullAction is set to WRAP.	

OE_TC_017				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
20. Verify that the *write_record* operation makes the records available for retrieval in the order received, even if the log has wrapped.	**Pass**: The *write_record* operation keeps track of the oldest log record and it is updated if the log is filled and the LogFullAction is set to WRAP. (OE0700-C235) **Fail**: There are any circumstances where *write_record* operation will place records in an order that will not result in the records being retrieved in order, beginning with the oldest record. (OE0700-C235)		Failure of this criterion OE0700-C235 means failure of the requirement OE0700.	
End of Test				

Test Recording Log - OE_TC_017					
DMD File					
Step 2 (Log Service)					
DCD File					
Steps 4 and 5 (Log Service)					
SPD File					
Step 6 (Log Service)		Steps 8 and 9 (*write_record* operation)	Steps 10 and 11 (LogFullAction operation)		

Test Summary: OE_TC_017

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C231 _____

OE0700-C232 _____

OE0700-C233 _____

OE0700-C234 _____

OE0700-C235 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.14. OE_TC_018 - OMG Lightweight Log Service :: clear_log

Test Case Number: OE_TC_018
OMG Lightweight Log Service::clear_log

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Specification
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification
	OMG Lightweight Log Specification
OE0700-C236	This operation(*clear_log()*) purges all logging records from the log storage area; however, it does not alter the size of the storage area in any way.
OE0700-C237	The log will set the availability status *logFull* state to false.
OE0700-C186	The *get_n_records()* operation returns the number of logging records currently stored in the log storage area.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2, Section 3.1.2.2
Lightweight Log Service Specification	February 2005 Version 1.1 formal/05-02-02	Section 3.3.4.4, Pages 3-25 – 3-26.

Test Objective

This test case verifies the log service requirement, OE0700, partially by testing the criteria, OE0700-C236 and OE0700-C237 of the OMG Lightweight Log. 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.

Places to Verify

Log Service

OMG Log Interfaces

Operations

void clearLog ();

unsigned long long get_n_records();

Preconditions

- All the Domain Profile files are available.
- The source code files are available.

Test Description

A. Determine from the Domain Profile whether log services are provided. (OE0700)
   1. **Untested:** No log services are provided.

B. Verify that the SPD files have the implementation declaration for the log services. (OE0700)
   1. **Pass:** The SPD file has the log service executable declaration.
   2. **Untested:** The SPD file has no log service executable declaration.

For each log service identified in Step A, perform the following steps:

C. Verify that the *clear_log()* operation removes all the logging records from the logging storage area, that the *clear_log()* operation sets the number of logging records contained in the logging storage area to zero, and that the *clear_log()* operation does not change the size of the logging storage area.(OE0700-C186, OE0700-C236)
   1. **Fail:** The *clear_log()* operation violates at least one of the following criteria: does not remove all the logging records from the storage area, does not set the number of logging records contained in the area to zero, changes the size of the logging storage area.
   2. **Pass:** The *clear_log()* operation successfully removes all the logging records from the storage area, sets the number of logging records to zero and the size of the logging storage area is not changed by this operation.

D. Verify that the get_n_records() operation returns the number of logging records currently stored in the log storage area (OE0700-C186)
   1. Fail: The get_n_records() operation returns the correct number of logging records..
   2. Pass: The get_n_records() operation fails to return the correct number of logging records..

E. Verify that the *clear_log()* operation sets the availability status, *logFull*, to false. (OE0700-C237)
   1. **Fail:** The *clear_log()* operation allows the availability status, *logFull*, to change or remain at the value of true.
   2. **Pass:** The *clear_log()* operation changes the availability status, *logFull*, to false.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested or N/A.
2. The Test Recording Log is intended to record results for data.

OE_TC_018				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine from the domain profile whether log services are provided (DMD file examination). (OE0700)				
NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). If the OE software engineer is not available, then steps 1 thru 7 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all the source code to find the operation. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.				
1. Locate and open the DMD file(s) for the Domain Manager.	DMD file opens. Note: The DMD file may be named *DomainManager.dmd.xml*			
2. Determine the log service(s) used, if any, the name of the service.	Log service(s) found. **Untested:** A log service is not found. (OE0700) Note: Most likely, the name of the log service is ***LogService***. In our example, we are using ***qqqqq***.		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> xxxxx </usesidentifier> <findby> <domainfinder **type**="**log**" **name**="**qqqqq**"> </domainfinder> </findby> </service> …</services>	
3. Locate and open the DCD file(s)	DCD file opens.		The Domain DCD file may be named DeviceManager.dcd.xml.	

OE_TC_018				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate log service *componentplacement* declaration in the DCD file using the name of the log service from the DMD and record the *componentfileref refid*.	*componentplacement* declaration for declared log service exists in the DCD file. **Untested**: *componentplacement* declaration for declared log service does not exist in the DCD file. (OE0700)		The name of the log service will be a usagename in the componentplacement section of the DCD. XML declaration example: <componentplacement> <componentfileref refid="**qqqqq_File**"> </componentfileref> <componentinstantiation id="**xxxxx**" <usagename>**qqqqq** </usagename> </componentinstantiation> </componentplacement>	
5. Locate the log service *componentfile id* declaration *using the componentfileref refid* and record the related SPD file name.	*Componentfile id* declaration for declared log service exists in the DCD file.		The refid from step 4 will be the id in the componentfile section of the DCD. XML declaration example: <componentfiles> <componentfile id=**"qqqqq_File"** type="Software Package Descriptor"> <**localfile name="qqqqq.spd.xml"/>** </componentfile>	
B. Verify that the SPD files have the implementation declaration for the log services. (OE0700)				

OE_TC_018				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Examine the SPD files (step 5) and record the local file name.	**Pass**: log service executable declaration exists.(OE0700) **Untested**: log service executable declaration does not exist.(OE0700)		XML declaration example: <implementation id="xxxx"> <code type= "xxxxx"> <**localfile name=** **"…/qqqqq.exe**"/>	
Note: the executable name is a clue to the source code. Conventional practice is to have the same name for the executable as the primary source code file. If the executable file name does not help				
C. Verify that the *clear_log()* operation removes all the logging records from the logging storage area, that the *clear_log()* operation sets the number of logging records contained in the logging storage area to zero, and that the *clear_log()* operation does not change the size of the logging storage area. (OE0700-C186, OE0700-C236)				
7. Verify that the *clear_log()* operation exists	**Pass:** The *clear_log()* operation exists. (OE0700-C236) **Fail:** The *clear_log()* operation does not exist. (OE0700-C236)		Failure of criterion OE0700-C236 means failure of the requirements OE0700.	
8. Examine the source code and verify that the *clear_log()* operation removes all the logging records from the logging storage area.	**Pass:** The *clear_log()* operation removes all the logging records from the logging storage area. (OE0700-C236) **Fail:** The *clear_log()* operation does not remove all the logging records from the logging storage area. (OE0700-C236)		Failure of criterion OE0700-C236 means failure of the requirements OE0700.	
9. Examine the source code and verify that the *clear_log()* operation does not change the size of the logging storage area.	**Pass:** The *clear_log()* operation does not change the size of the logging storage area. (OE0700-C236) **Fail:** The *clear_log()* operation changes the size of the logging storage area. (OE0700-C236)		Failure of criterion OE0700-C236 means failure of the requirements OE0700.	
D. Verify that the *get_n_records()* operation returns the number of logging records currently stored in the log storage area (OE0700-C186)				

OE_TC_018				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. Examine the source code and verify that the *get_n_records()* operation returns zero after the *clear_log()* operation is performed.	**Pass:** The *get_n_records()* operation should accurately report that the number of logs contained in the logging storage area is zero. (OE0700-C186) **Fail:** The *get_n_records()* operation does not accurately report that the number of logs contained in the logging storage area is zero (that is non-zero is reported). (OE0700-C186)		The source code may actually count the number of logs in the logging storage area at execution time or have a variable to store the number of records when the operation *get_n_records()* is called. the *get_n_records()* operation retrieves the number of logging records currently stored in the log. Both the variable counter and the actual number of records in the logging area should be checked. Failure of this criterion OE0700-C186 means failure of the requirements OE0700.	
E. Verify that the *clear_log()* operation sets the availability status, *logFull*, to false. (OE0700-C237)				
11. Examine the source code and verify that the *clear_log()* operation sets the availability status, *logFull*, to false.	**Pass:** The *clear_log()* operation sets the availability status, *logFull*, to false. (OE0700-C237) **Fail:** The *clear_log()* operation does not set the availability status, *logFull*, to false. (OE0700-C237)		Testers should check whether there is always a logic inside clear_log() to set the logFull to true. Such logic is optional when the log's logFullAction is WRAP instead of HALT. Failure of criterion OE0700-C237 means failure of the requirements OE0700.	
End of Test				

Test Recording Log - OE_TC_018							
DMD file							
Step 2 Log Service							
DCD file							
Step 4 Log Service							
SPD file							
Step 6 Log Service executable		Step 7 and 8 clear_log operation		Step 8 get_n_records operation		Step 9 clear_log operation	
Step 10 and 11 clear_log operation							

Test Summary: OE_TC_018

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C236 _____

OE0700-C237 _____

OE0700-C186 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.15. OE_TC_019 - OMG Lightweight Log Service :: destroy

Test Case Number: OE_TC_019

OMG Lightweight Log Service::destroy

Requirements

SCA v2.2.2 Tag	OMG Lightweight log
OE0700	If a log service is implemented, the log service shall conform to the OMG Lightweight Log Service Specification destroy the associated instance of the Log class.
	OMG Lightweight Log Specification
OE0700-C238	This operation will destroy the associated instance of the Log class.
OE0700-C239	All existing records in the log storage area are irrecoverably lost and the memory resources associated with the storage area are released.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 05-15-2006	Page 3-2
Lightweight Log Service Specification	February 2005, Version 1.1, formal/05-02-02	Page 2-22 (Section 2.6.5), Page 3-22, Page 3-26 (Section 3.3.4.5)

Test Objective

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.

Places to Verify

Log Service

OMG Log Interfaces

Operations

void destroy ();

Preconditions

- The Domain Profile and source code files are available.

Test Description

A. Determine whether log services are provided (OE0700, OE0700-C238, OE0700-C239)
   1. **Untested:** No log service is provided
B. For each log service identified in Step A, perform the following steps:
   1. Verify that the instance of the log class is destroyed (OE0700-C238)
      a. **Pass:** The proper log class instance is destroyed
      b. **Fail:** The log class is not destroyed
   2. Verify that all memory associated with the log storage area is released. (OE0700-C239)
      a. **Pass:** All memory associated with the log storage area is released
      b. **Fail:** All memory associated with the log storage area is not released

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record long comments and list other file/data.

OE_TC_019				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine whether log services are provided. (OE0700)				
NOTE: The OE software engineer can be of assistance in locating the source code for the log service(s). Steps 1 thru 5 provide some ideas to help in locating the source code. A search engine such as grep or a development tool such as Visual C++ can be used to search all source. Do not depend on Microsoft explorer's search operation, as it can be demonstrated that it will not find a string within a file.				
1. Locate and open the DMD file(s) for the OE.	DMD file opens.			
2. Determine the log service used, if any, recording the name of the service.	**Untested:** A log service is not found. (OE0700)		A log service can be identified in a DMD file by the following hierarchy: <services> … <service> <usesidentifier> Xxx </usesidentifier> <findby> <domainfindertype="**log**" name="**qqqqqq**"> </domainfinder> </findby> </service> … </services>	
3. Locate and open the DCD file(s) for the OE.	DCD file opens.			

OE_TC_019				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Locate the log service from the DMD in the DCD file. Record the SPD file for the log service.	**Untested:** A log service is not found. (OE0700)		First, using the name from the DMD file, find in the DCD file: componentinstantiation refid using the following hierarchy: <componentplacement> <componentfileref refid="**rrrrr**"> </componentfileref> <componentinstantiation id="xxxx"> <usagename>**qqqqqq** </usagename> </componentinstantiation> </componentplacement> Then find the SPD file using the componentinstantiation id and the following hierarchy: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name= "**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	
5. Identify any other log services provided in the DCD file and record the SPD files.	All other log services are located.		Log service in DCD: <connections> <connectinterface id="xxxxxx"> <usesport>	

OE_TC_019				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
			<usesidentifier>LogPort</usesidentifier> <findby> <namingservice name="yyyyyy"> </namingservice> </findby> </usesport> <findby> <domainfinder type="**log**" name="**nnnn**"> </domainfinder> </findby> </connectinterface> </connections> SPD file: <componentfiles> … <componentfile id="**rrrrr**" type="SPD"> <localfile name="**ssssss**.spd.xml"> </localfile> </componentfile> </componentfiles>	
For each log service identified, perform the following steps **B.1 Verify that the instance of the log is destroyed. (OE0700-C238)**				

OE_TC_019				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Determine and open the source code file for the log service implementation containing the destroy() operation.	**Pass:** Log service implementation is found. (OE0700-C238) **Fail:** Log service implementation is not found. (OE0700-C238)		The name of the executable in the SCD file is the best clue for the name of the source code. Failure of criterion OE0700-C238 means failure of the requirements OE0700.	
7. Find the destroy operation in the source code.	**Pass:** Source code having the *destroy* operation is found. (OE0700-C238) **Fail:** Source code having the *destroy* operation is not found. (OE0700-C238)		A search tool such as grep or a GUI development tool such as Visual C++ will simplify this. Failure of criterion OE0700-C238 means failure of the requirements OE0700.	
8. Verify that the log class is destroyed as a result of the destroy() operation being called.	**Pass:** The log class is destroyed as a result of the *destroy* operation being called. (OE0700-C238) **Fail:** The log class is not destroyed as a result of the *destroy* operation being called. (OE0700-C238)		If the destructor for the class is called, this would be sufficient. Failure of criterion OE0700-C238 means failure of the requirements OE0700.	
B.2 Verify that all memory associated with the log storage area is released. (OE0700-C239)				
9. Verify that all memory associated with the log storage area is released.	**Pass:** All memory associated with the log storage area is released. (OE0700-C239) **Fail:** All memory associated with the log storage area is not released. (OE0700-C239)		Failure of criterion OE0700-C239 means failure of the requirements OE0700.	
End of Test				

DMD file				
Step2 (log service)	**Step3 (DCD file)**	**Step4 (log service)**	**Step5 (other log services)**	**Step6 (source code file)**
Step8 (name of destructor class)	**Step9 (constructor class for memory allocations)**			

Test Summary: OE_TC_019

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s) (x). Failure of any OE0700 criteria results in a failure of OE0700.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0700 _____

OE0700-C238 _____

OE0700-C239 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

B.1.16. OE_TC_020 - Port :: connectPort

Test Case Number: OE_TC_020

Port::connectPort

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0069	The *connectPort* operation shall make a connection to the component identified by its input parameters.
OE0069-C002	A port may support several connections. The input connectionId is a unique identifier to be used by the *disconnectPort* operation when breaking a specific connection.
OE0070	The *connectPort* operation shall raise the InvalidPort exception when the input connection parameter is an invalid connection for this port.
OE0070-C004	The InvalidPort exception indicates one of the following errors has occurred in the specification of a *Port* association: 1. errorCode 1 means the *Port* component is invalid (unable to narrow object reference) or illegal object reference,

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-8, Section 3.1.3.1.1.5.1.3 ; Page 3-8, Section 3.1.3.1.1.5.1.5 ; Page 3-7 Section 3.1.3.1.1.3.1
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-13, Section C.1

Test Objective

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.

Places to Verify

Devices, resource and services or any other components that implement the *Port* interface.

IDL References

Exceptions

exception InvalidPort { unsigned short errorCode; string msg; };
exception OccupiedPort { };

Operations

void *connectPort* (
in Object connection,
in string connectionId)
raises (
CF::Port::InvalidPort, CF::Port::OccupiedPort);

Preconditions

- The source code files of the Core Framework (CF) are available.
- CF documentation is available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate all implementations of the *connectPort* operation in the Core Framework source code. (OE0069, OE0070)
   1. **N/A**: There are no implementations of the *connectPort* operation in the Core Framework source code.

For each implementation of the *connectPort* operation found in the Core Framework source code, perform the following steps:

B. Verify that the *connectPort* operation only attempts connections to a valid input object from its parameter and the *connectPort* operation associates the input connectionId with this input connection object parameter if the attempt is successful. (OE0069, OE0069-C002)
   1. **Pass**: The *connectPort* operation only attempts connections to a valid input object from its parameter and the *connectPort* operation associates the input connectionId with this connection if the attempt is successful.
   2. **Fail**: The *connectPort* operation attempts a connection to an input object from its parameter that is not valid.
   3. **Fail**: The *connectPort* operation does not associate the input connectionId with the connection if the attempt is successful.

C. Verify that the *connectPort* operation raises the InvalidPort exception with an error code of 1 when the object reference from the operation parameter cannot be narrowed to a valid component type. (OE0070, OE0070-C004)
   1. **Pass**: The *connectPort* operation raises the InvalidPort exception with an errorCode of 1 when it cannot narrow the object reference to a valid component type.

2. **Fail**: The *connectPort* operation does not raise the InvalidPort exception and/or it does not contain an errorCode of 1 when the operation cannot narrow the object reference to a valid component type.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_020				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate all implementations of the *connectPort* operation in the Core Framework source code. (OE0069, OE0070)				
1. Locate all implementations of *connectPort* in the Core Framework source code and record the source files containing the implementations.	**N/A**: There are no implementations of the *connectPort* operation in the Core Framework source code. (OE0069, OE0070)			
For each implementation of the *connectPort* operation found in the Core Framework source code, perform the following steps:				
B. Verify that the *connectPort* operation only attempts connections to a valid input object from its parameter and the *connectPort* operation associates the input connectionId with this input connection object parameter if the attempt is successful. (OE0069, OE0069-C002)				
2. Verify in the Core Framework source code that the *connectPort* operation establishes the connection based on the input connection object parameter.	**Pass**: The *connectPort* operation establishes this connection based on the input connection object parameter. (OE0069) **Fail**: The *connectPort* operation does not establish this connection based on the input connection object parameter. (OE0069)			

OE_TC_020				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify in the source code that the *connectPort* operation associates the connectionId with the connection established between the object that performs the *connectPort* operation and the object represented in its parameter.	**Pass**: The *connectPort* operation associates the input connectionId with this connection if the attempt is successful. (OE0069-C002) **Fail**: The *connectPort* operation does not associate the input connectionId with the connection if the attempt is successful. (OE0069-C002**)**		These objects represent uses and provides ports of the component. The connectionId is used by the disconnectPort operation to correctly distinguish and disconnect the port. Failure of the criterion, OE0069-C002, means failure of the requirement OE0069.	
C. Verify that the *connectPort* operation raises the InvalidPort exception with an error code of 1 when the object reference from the operation parameter cannot be narrowed to a valid component type. (OE0070, OE0070-C004)				
5. Verify in the *connectPort* source code that when it cannot narrow the object reference to a valid component type, the operation raises an InvalidPort exception.	**Pass**: The *connectPort* operation raises the InvalidPort exception when it cannot narrow the object reference to a valid component type. (OE0070) **Fail**: The *connectPort* operation does not raise the InvalidPort exception when the operation cannot narrow the object reference to a valid component type. (OE0070)			

OE_TC_020				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
6. Verify in the *connectPort* source code that when the operation raises an InvalidPort exception, it containing an errorCode value of 1.	**Pass**: The *connectPort* operation raises an InvalidPort exception that contains an errorCode value of 1. (OE0070-C004) **Fail**: The *connectPort* operation raises an InvalidPort exception that does not contain an errorCode value of 1. (OE0070-C004)		Failure of this criterion OE0070-C004 means failure of the requirement OE0070.	
End of Test				

Test Recording Log – OE_TC_020				
Step 1 (source files of the *connectPort* operation)	Step 2 (input object used for connection) (Pass/Fail?)	Step 3 (Connection associated with input id) (Pass/Fail?)	Step 4 (invalid object reference raises exception) (Pass/Fail?)	Step 5 (error code must be 1.) (Pass/Fail?)

Test Summary OE_TC_020

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

Pass:	No failures detected
Fail:	Failure(s) detected in Step(s)(x). Failure of the associated criterion, OE0069-C002, results in a failure of requirement OE0069. Failure of the associated criterion, OE0070-C004, results in a failure of requirement OE0070.
Untested:	Condition which is not testable
N/A:	Not Applicable

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

OE0069_____

OE0069-C002 _____

OE0070_____

OE0070-C004 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.17. OE_TC_032 - Port :: connectPort raises OccupiedPort

Test Case Number: OE_TC_032

Port::connectPort raises OccupiedPort exception

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0071	The *connectPort* operation shall raise the OccupiedPort exception when unable to accept the connections because the port is already fully occupied.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-8, Section 3.1.3.1.1.5.1.5; Page 3-7, Section 3.1.3.1.1.3.2
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-13, Section C.1

Test Objective

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.

Places to Verify

Components of the Core Framework that implement/use the *Port* interface, which may include one or more of the following: devices, resources, device managers and/or services.

IDL References

Exceptions

exception OccupiedPort { };

Operations

void *connectPort* (in Object connection, in string connectionId) raises (CF::Port::InvalidPort, CF::Port::OccupiedPort);

Preconditions

- The Core Framework source code files are available.

- The Domain Profile of the Core Framework is available
- The Core Framework documentation is available.
- The passing of test procedure, OE_TC_102 (Base Application Interfaces).
- The passing of test procedure, OE_TC_119 of the Core Framework.

Test Description

A. Obtain the list of ports implemented in the Core Framework from test procedure OE_TC_119. (OE0071)
   1. **N/A**: A component of the Core Framework does not use the *Port* interface.

For each port from the list obtained in step A, perform the following steps:

B. Identify the implementation of the *connectPort* operation (OE0071).
   1. **Pass:** All ports contain the *connectPort* operation.
   2. **Fail:** The port does not contain the *connectPort* operation.

For each implementation of the *connectPort* operation, perform the following steps.

C. Verify that the *connectPort* operation checks whether the port is fully occupied before making the connection and raises the *OccupiedPort* exception if a connection is attempted when the port is fully occupied (OE0071).
   1. **Pass**: The *connectPort* operation raises the *OccupiedPort* exception if a connection is attempted when the port is fully occupied by reaching the threshold of allowed connections.
   2. **Fail**: The *connectPort* operation does not raise the *OccupiedPort* exception if a connection is attempted when the port is fully occupied by reaching the threshold of allowed connections.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_032				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Obtain the list of ports implemented in the Core Framework from test procedure OE_TC_119. (OE0071)				
1. From the results of OE_TC_119, obtain the list of ports implemented in the Core Framework.	The ports of the Core Framework are provided in a list by test procedure OE_TC_119. **N/A**: A component of the Core Framework does not contain uses ports. (OE0071)		In most cases, these will be represented as the components uses ports. OE_TC_119 uses an automated tool to examine the Domain Profile files of the Core Framework.	
For each port from the list of ports generated from OE_TC_119, perform the following step:				
B. Identify the implementation of the *connectPort* operation. (OE0071)				
2. Examine the Core Framework source code of the devices and components that are represented in the list provided in step A and identify the implementations of the *connectPort* operation. List the files containing the *connectPort* operation.	**Pass:** All found ports contain the *connectPort* operation. (OE0071) **Fail:** One or more ports do not contain the *connectPort* operation. (OE0071)		Note: Port components may inherit the implementation from a parent class. Clues to the names of the source code files of the ports may be similar to the names of the ports themselves because the Spectra tool uses extracts the names from the Domain Profile. Another way to determine the source code of the ports is to examine Core Framework documentation and talk to the developer.	
For each implementation of the *connectPort* operation, perform the following step:				
C. Verify that the connectPort operation checks whether the port is fully occupied before making the connection and raises the OccupiedPort exception if a connection is attempted when the port is fully occupied. (OE0071)				

OE_TC_032				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Examine the source code of the *connectPort* operation and determine whether the *connectPort* operation checks the limit of connections to ascertain if the port is fully occupied.	The *connectPort* operation checks whether the port is fully occupied before making the connection.		In this step "fully occupied" means reaching a predefined limit to the number of connections it as able to process(between 1…n connections, with n the predefined limit)	
4. Verify in the *connectPort* operation source code that logic exists to raise the *OccupiedPort* connection when it detects that the port is and has reached the threshold of the number of connections allowed and is fully occupied.	**Pass**: The *connectPort* operation raises the *OccupiedPort* exception if a connection is attempted when the port is fully occupied by reaching the threshold of allowed connections. (OE0071) **Fail**: The *connectPort* operation does not raise the *OccupiedPort* exception if a connection is attempted when the port is fully occupied by reaching the threshold of allowed connections. (OE0071)			
End of Test				

Test Recording Log – OE_TC_032				
Step 1 (list of port names)	Step 2 (connectPort operation exists?)	Step 2 (file name(s) of *connectPort* operation)	Step 3 (checks limit of connections)	Step 4 (OccupiedPort exception)

Test Summary OE_TC_032

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

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

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

OE0071_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.18. OE_TC_035 - Port :: disconnectPort

Test Case Number: OE_TC_035

CF::Port::disconnectPort raises InvalidPort

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0072	The *disconnectPort* operation shall break the connection to the component identified by the input connectionId parameter.
OE0073	The *disconnectPort* operation shall raise the *InvalidPort* exception when the input connectionId parameter is not a known connection to the *Port* component.
OE0073-C003	The *InvalidPort* exception indicates one of the following errors has occured in the specification of a *Port* association: errorCode 2 means the *Port* name is not found (not used by this Port).

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-7, Section 3.1.3.1.1.3.1 Page 3-8, Section 3.1.3.1.1.5.2.3 and 3.1.3.1.1.5.2.5

Test Objective

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.

Places to Verify

Source Code Files with *Port disconnectPort* operation

IDL References

Exceptions

The InvalidPort exception indicates one of the following errors has occurred in the specification of a Port association:

- errorCode 1 means the *Port* component is invalid (unable to narrow object reference) or illegal object reference,
- errorCode 2 means the *Port* name is not found (not used by this Port).

exception InvalidPort { unsigned short errorCode; string msg; };

Operations

void disconnectPort (in string connectionId) raises (InvalidPort);

Preconditions

- All of the source code files for the OE Port component under test are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Verify that the source code for the *Port disconnectPort* operation is found. (OE0072)
   1. **Pass**: The source code for the *Port disconnectPort* operation is found.
   2. **Fail:** The source code for the *Port disconnectPort* operation is not found.

For each instance of a *disconnectPort* operation found:

B. Verify that the *InvalidPort* exception is raised, when the input connectionId parameter is not a known connection to the Port component. (OE0073)
   1. **Pass:** The *InvalidPort* exception is raised when the input connectionId parameter is not a known connection to the Port component.
   2. **Fail:** The *InvalidPort* exception is not raised when the input connectionId parameter is not a known connection to the Port component.

C. Verify the presence of an error code value of 2 when the *Port* name is not found. (OE0073-C003)
   1. **Pass:** The *InvalidPort* exception has an error code value of 2 when the *Port* name is not found.
   2. **Fail:** The *InvalidPort* exception does not have an error code value of 2 when the *Port* name is not found.

D. Verify that when no exception is raised, the port identified by connectionId is successfully disconnected using the *disconnectPort* operation (OE0072)
   1. **Pass:** The port identified by connectionId is successfully disconnected using the *disconnectPort* operation.
   2. **Fail:** The port identified by connectionId is not successfully disconnected using the *disconnectPort* operation.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_035				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Verify that the source code for the *Port disconnectPort* operation is found. (OE0072, OE0073)				
1. Identify and record the source code for the Port component.	**Pass:** The Port component is found. (OE0072, OE0073)			
2. Locate and record the file name of the source code for the *disconnectPort* operation of the Port component.	**Pass:** File is found. (OE0072, OE0073) **Fail**: File is not found. (OE0072, OE0073)		void disconnectPort (in string connectionId) raises (InvalidPort);	
For each instance of the *disconnectPort* operation found:				
B. Verify that the *InvalidPort* exception is raised, when the input connectionId parameter is not a known connection to the Port component. (OE0073)				
3. Verify that the *InvalidPort* exception is raised, when the input connectionId parameter is not a known connection to the Port component.	**Pass:** The *InvalidPort* exception is raised when the input connectionId parameter is not a known connection to the Port component. (OE0073) **Fail:** The *InvalidPort* exception is not raised when the input connectionId parameter is not a known connection to the Port component. (OE0073)			
C. Verify the presence of an error code value of 2 when the *Port* name is not found. (OE0073-C003)				

OE_TC_035				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Verify that the *InvalidPort* exception error code is 2, when the Port name is not found.	**Pass:** The *InvalidPort* exception has an error code value of 2 when the Port name is not found. (OE0073-C003) **Fail:** The *InvalidPort* exception does not have an error code value of 2 when the Port name is not found. (OE0073-C003)		Failure of this criterion OE0073-C003 means failure of the requirements OE0073. The error code of 2 may only be used in the exception being raised from a *disconnectPort* operation. The error code of 1 may only be used in the exception being raised from a *connectPort* operation.	
D. Verify that when no exception is raised, the port identified by connectionId is successfully disconnected using the *disconnectPort* operation. (OE0072)				
5. Verify that the *disconnectPort* operation uses the connectionId to identify the port to disconnect.	**Pass**: The port identified by the connectionId value is the one being disconnected. (OE0072) **Fail**: The port identified by the connectionId value is not the one being disconnected. (OE0072)			
6. Verify that the identified port is successfully disconnected using this operation.	**Pass**: The identified port is disconnected using this operation. (OE0072) **Fail**: The identified port is not disconnected using this operation. (OE0072)			
End of Test				

Test Recording Log – OE_TC_035					
Step 1 (Source Code for Port component)	Step 2 (Source Code for disconnect operation)	Step 3 (Unknown connectionId raises exception) Y/N	Step 4 (error code of 2 if Port id not valid) Y/N	Step 5 (port id's with connectionId is used by operation) Y/N	Step 6 (port can be disconnected with given command) Y/N

Test Summary OE_TC_035

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

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

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

OE0072_____

OE0073_____

OE0073-C003 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.19. OE_TC_037 - LifeCycle :: initialize raises InitializeError

Test Case Number: OE_TC_037

LifeCycle: initialize raises InitializeError exception

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0074	The *initialize* operation **shall** raise an *InitializeError* exception when an initialization error occurs.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-9, Section 3.1.3.1.2.3.1, Page 3-10, Section 3.1.3.1.2.5.1.5
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-14

Test Objective

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.

Places to Verify

Devices, CF Applications, and any other components with initialize methods (derived from LifeCycle).

IDL References

Operations

void initialize () raises (CF::LifeCycle::InitializeError);

Exceptions

exception InitializeError { StringSequence errorMessages; };

Preconditions

- The source code files of the Core Framework are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Identify the source code files that implement the *LifeCycle::initialize* operation. (OE0074)
   1. **Untested:** The source code files are not available.

For each implementation of the *LifeCycle::initialize* operation within the OE, perform the following steps:

B. Verify that the *LifeCycle::initialize* operation raises the *InitializeError* exception where an error occurred during component initialization. (OE0074)
   1. **Pass:** The *LifeCycle::initialize* operation raises the *initializeError* exception.
   2. **Fail:** The *LifeCycle::initialize* operation does not raise the *initializeError* exception.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_037				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement the *LifeCycle::initialize* operation. (OE0074)				
1. Perform a keyword search for the *LifeCycle::initialize* operation on all source code provided by the developer.	**Untested:** No results from keyword search. (OE0074)		May need the help of the software engineer to locate the source code files directories.	
2. Examine the source code files returned in Step 1 and search for the *LifeCycle::initialize* operation implementation. Record the file name.	Each implementation of the *LifeCycle::initialize* operation is identified and recorded.		When searching for an implementation of the *initialize* operation, make sure the class that implements the function directly or indirectly inherits from the LifeCycle class. Ignore any other non-LifeCycle implementation.	
For each implementation of the *LifeCycle::initialize* operation within the OE, perform the following steps:				
B. Verify that the *LifeCycle::initialize* operation raises the *initializeError* exception where an error occurred during component initialization. (OE0074)				
3. Search within the *LifeCycle::initialize* operation where an error occurred during component initialization. Verify that each possible error will eventually lead to the throwing of the InitializeError.	**Pass**: The *LifeCycle::initialize* operation raises the *initializeError* exception. (OE0074) **Fail**: The *LifeCycle::initialize* operation does not raise the *initializeError* exception. (OE0074)		Source code example: void PseudoDevice_Device_i::initialize () ACE_THROW_SPEC ((CORBA::SystemException, CF::LifeCycle::InitializeError))))	
End of Test				

Test Recording Log – OE_TC_037		
Step 2 (source files with *LifeCycle::initialize* operation)	**Step 3** (source files with *InitializeError* exception)	**Notes**

Test Summary OE_TC_037

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

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

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

OE0074_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.20. OE_TC_038 - LifeCycle :: releaseObject

Test Case Number: OE_TC_038

LifeCycle::releaseObject

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0075	The *releaseObject* operation shall release all internal memory allocated by the component during the life of the component.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Pages 3-10, Section 3.1.3.1.2.5.2

Test Objective

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.

Places to Verify

Resources, Devices, Applications, and any other components with releaseObject methods

IDL References

Operations

void releaseObject () raises (CF::LifeCycle::ReleaseError);

Preconditions

- The CF source code files are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate all memory allocations with the CF source code. (OE0075)
   1. **N/A:** No memory allocations are found.
B. Locate all memory deallocations with the CF source code. (OE0075)

1. **N/A:** No memory allocations and no memory deallocations are found.
2. **Fail**: Memory allocations were found in step A and no memory deallocations are found.

C. Verify that every memory allocation has a corresponding memory deallocation. (OE0075)
1. **Pass**: All memory allocations found have corresponding memory deallocations.
2. **Fail**: At least one memory allocation is found that does not have a corresponding memory deallocation.
3. **Fail**: There is a memory deallocation that does not have a corresponding memory allocation.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_038				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate all memory allocations with the CF source code. (OE0075)				
1. Locate all instances of memory allocations and record the file and line number.	**N/A**: No memory allocations are found. (OE0075)		This means searching the source code for new, malloc, calloc and any other functions that are used to allocate memory in the programming language use. An Integrated Development Environment (IDE) is the best tool for this search.	
B. Locate all memory deallocations with the CF source code. (OE0075)				
2. Locate all instances of memory deallocations and record the file and line number.	**N/A**: No memory deallocations are found. (OE0075) **Fail**: Allocations were found in step 1 but no memory deallocations are found. (OE0075)		This means searching the source code for delete, free, and any other functions that are used to deallocate memory in the programming language use. An Integrated Development Environment (IDE) is the best tool for this search.	
C. Verify that every memory allocation has a corresponding memory deallocation. (OE0075)				

OE_TC_038				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that each allocation found in step 1 has a corresponding deallocation that was found in step 2.	**Pass**: Every allocation has a corresponding deallocation and every deallocation has a corresponding allocation. (OE0075) **Fail**: There is at least one allocation that does not have a corresponding deallocation. (OE0075) **Fail**: There is at least one deallocation that does not have a corresponding allocation. (OE0075)			
End of Test				

Test Recording Log – OE_TC_038				
Step 1 (allocation – file and line number)	Step 2 (deallocation – file and line number)	Step 3 (allocation/deallocation paired)	Step 4 (constructer allocates)	Step 5 (destructor deallocates)

Test Summary OE_TC_038

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

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

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

OE0075_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.21. OE_TC_039 - LifeCycle :: releaseObject raises ReleaseError

Test Case Number: *OE_TC_039*

CF::LifeCycle::releaseObject raises ReleaseError

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0078	The *releaseObject* operation shall raise a *ReleaseError* exception when a release error occurs.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-9, Section 3.1.3.1.2.3.2, Page 3-10, Section 3.1.3.1.2.5.2.5
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-14

Test Objective

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

Places to Verify

Resources, Devices, Applications, and any other components with CF::LifeCycle::*releaseObject* methods

IDL References

Exceptions

exception ReleaseError { CF::StringSequence errorMessages; };

Operations

void releaseObject ()
raises (CF::LifeCycle::ReleaseError); };

Preconditions

- The source code of the Core Framework is available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Identify the source code files that implement the CF::LifeCycle::*releaseObject* operation in the source code. (OE0078)
   1. **Pass:** An implementation of the CF::LifeCycle::*releaseObject* operation is found.
   2. **Fail:** An implementation of the CF::LifeCycle::*releaseObject* operation is not found.

For each CF::LifeCycle::*releaseObject* operation found within the OE under test, perform the following steps:

B. Verify if a release error occurs, the CF::LifeCycle::*ReleaseError* exception will be raised. (OE0078)
   1. **Pass:** The CF::LifeCycle**::***ReleaseError* exception is raised for an occurrence of a release-related error.
   2. **Fail:** The CF::LifeCycle**::***ReleaseError* exception is not raised for an occurrence of a release-related error.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_039				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement the CF::LifeCycle::*releaseObject* operation. (OE0078)				
1. Perform a search for the CF::LifeCycle**::***releaseObject* operation on source code provided by the developer.	**Untested:** The source code files of the CF::LifeCycle**::***releaseObject* operation are not available. **(**OE0078)		May need the help of the software engineer to locate the source code files. A keyword search is another method to locate the operation in the source code files.	
2. Examine the source code files returned in Step 1 and search for the implementation of the CF::LifeCycle**::***releaseObject* operation. Record the file name.	The CF::LifeCycle**::***releaseObject* operation for each implementation is identified and recorded.			
For each CF::LifeCycle::*releaseObject* operation found within the OE under test, perform the following steps:				
B. Verify if a release error occurs, the CF::LifeCycle::*ReleaseError* exception will be raised. (OE0078)				
3. Verify in the CF::LifeCycle::*releaseObject* operation code that if a release-related error occurs, then the CF::LifeCycle::*ReleaseError* exception will be raised.	**Pass:** The CF::LifeCycle::*ReleaseError* exception is raised for an occurrence of a release-related error. (OE0078) **Fail:** The CF::LifeCycle::*ReleaseError* exception is not raised for an occurrence of a release-related error. (OE0078)			
End of Test				

Test Recording Log OE_TC_039	
Step 2 (source file with CF::LifeCycle::releaseObject)	Step 3 *(*CF::LifeCycle::*ReleaseError* exception raised?)

Test Summary OE_TC_039

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

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

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

OE0078_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.22. OE_TC_041 - TestableObject :: runTest uses testId parameter

Test Case Number: OE_TC_041

TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0079	The *runTest* operation shall use the input *testId* parameter to determine which of its predefined test implementations should be performed.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1.3

Test Objective

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.

Places to Verify

Devices, CF Applications, and any other components with *runTest* methods (derived from TestableObject).

IDL References

Exceptions

exception UnknownTest { };
exception UnknownProperties { CF::Properties invalidProperties; };

Operations

void runTest (in unsigned long testId, inout CF::Properties testValues)
raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);

Preconditions

- All the Domain profile files are available.

- The TestableObjects (e.g. Devices, CF Applications) source code files are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Identify all Devices and components that support the *runTest* operation. (OE0079)
   1. **N/A**: No devices or components support the *runTest* operation
B. Verify that the input *testId* parameter is used to determine which predefined test to run. (OE0079)
   1. **Pass:** The input *testId* parameter determines which predefined test to run.
   2. **Fail:** The code does not use the input *testId* parameter to determine which predefined test to run.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_041				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify all Devices and components that support the RunTest operation.				
1. From the source code, Identify all devices and components that inherit the type Resource or inherit TestableObject directly. Record the names of the source files containing the implementation of the *runTest* operation.	**N/A**: No devices or components support the *runTest* operation. **(**OE0079)			
B. Verify that the input *testId* parameter is used to determine which predefined test to run. (OE0079)				
2. From the source code, verify that the input *testId* parameter is used to determine which predefined test to run.	**Pass**: The input *testId* parameter determines which predefined test to run. (OE0079) **Fail**: The code does not use the input *testId* parameter to determine which predefined test to run. (OE0079)			
End of Test				

Step 1 (source file names)	**Step 2** (testId determines predefined tests (Y/N ?)

Test Summary OE_TC_041

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

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

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

OE0079_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.23. OE_TC_042 - TestableObject :: runTest uses testValues parameter

Test Case Number: OE_TC_042
TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0080	The id/value pair(s) of the testValues parameter shall be used to provide additional information to the implementation-specific test to be run.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1.3
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-15, Section C.1
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-24 thru D-25, Section D.4.1.3

Test Objective

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.

Places to Verify

Devices, CF Applications, and any other components with *runTest* methods (derived from *TestableObject*).

IDL References

Exceptions

CF::TestableObject::UnknownTest,
CF::UnknownProperties;

Operations

void runTest (in unsigned long testId, inout CF::Properties testValues)
raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);

Preconditions

- The Testable Objects (e.g. Devices, CF Applications) source code files are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate the source code of devices/components that implement the *RunTest* operation. (OE0080).
   1. **N/A**: The *RunTest* is not implemented for any devices/components. End test.

B. Verify the association between the testValues parameter and the implementation-specific runTest() operation as the way to provide additional test information. (OE0080).
   1. **Pass:** The testValues parameter is used to provide the additional information (input values and/or expected results) for the particular test, if it requires such additional information.
   2. **Fail:** The testValues parameter is not used to provide the additional information (input values and/or expected results) for the particular test, if it requires such additional information.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_042				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code for devices/components that implement the *RunTest* operation. (OE0080)				
1. Locate the implemented *runTest()* source code for devices/components.	**N/A**: The *RunTest* operation is not implemented for any devices/components. End test. (OE0080).		runTest is derived from TestableObject	
Note: For each device/component that implements the *runTest* operation, determine, perform the following:				
B. Verify the association between the testValues parameter and the implementation-specific *runTest()* operation as the way to provide additional test information. (OE0080)				
2. Look at the PRF file(s) and determine if it defines any 'test' element kind.	**N/A**: No test has been implemented/defined by the developer. Continue with the next component. (OE0080)			
3. For each device/component determine if the testId(s) were/was found in the XML file.	**N/A**: testId was not found in the XML code. Continue with the next component. (OE0080). **Pass:** testId(s) were/was found in the XML code. (OE0080)		The testId is the link to the test properties for the additional test information. For example, there can be multiple tests for a device and, therefore, there would be multiple testIds.	

OE_TC_042				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. In the XML file, search for and list the inputvalue(s) and resultvalue(s) element(s) corresponding to each testId.	**N/A**: inputvalue(s) and resultvalue(s) elements were not found in the XML code. (OE0080). Continue with the next testId. **Pass**: inputvalue(s) and resultvalue(s) elements were found in the XML code. (OE0080)		The test element contains inputvalue and resultvalue elements and is used to specify a list of test properties for executing the runTest() operation in order to perform a component-specific test.	
5. If the inputvalue is defined in the XML file for this specific testId, verify that the testValues parameter is parsed in the *runTest()* operation to retrieve any input value. (If inputvalue is not defined for this specific testId in the XML, then continue to the next step.)	**Pass**: testValue is used to retrieve any input value that is part of the design to provide the additional information. (OE0080) **Fail**: testValue is not used to retrieve any input value that is part of the design to provide the additional information. (OE0080)		SCA requires one resultvalue, but there is no SCA requirement to even have any inputvalues. The inputvalue element would be used to configure the test to be performed.	
6. If the resultvalue is defined in the XML file for this specific testId, verify that the testValues parameter is parsed in the *runTest()* operation to retrieve any resultvalue.	**Pass**: testValues is used to retrieve any resultvalue, which is part of the design to provide additional test information. (OE0080). **Fail**: testValues is not used to retrieve any resultvalue, which is part of the design to provide additional test information (OE0080).		It is SCA-legal to have a resultvalue, while not having an inputvalue. The resultvalues is used to specify the desired results of the *runTest* operation.	
End of Test				

Test Recording Log – OE_TC_042				
Step1 (Device/Component implementing runTest())	**Step3** (List all testId(s) found in the XML for each device/component in step 1)	**Step4** (List inputvalue and resultvalue for each testId in the XML that were found)	**Step5** (The inputvalue is parsed if the third column in this table is not empty for the specific testId?) (Yes / No)	**Step6** (The resultvalue is parsed if the third column in this table is not empty for the specific testId?) (Yes / No)

Test Summary OE_TC_042

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

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

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

OE0080_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.24. OE_TC_043 - TestableObject :: runTest returns results

Test Case Number: OE_TC_043

TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0081	The *runTest* operation shall return the result(s) of the test in the testValues parameter.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1.3
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Page C-15, Section C.1

Test Objective

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.

Places to Verify

Devices and any other components with *runTest* methods (derived from TestableObject).

IDL References

Data

```
typedef sequence <DataType> Properties;
struct DataType { string id; any value;};
```

Exceptions

```
exception UnknownTest { };
exception UnknownProperties { CF::Properties invalidProperties; };
```

Operations

```
void runTest ( in unsigned long testId, inout CF::Properties testValues )
        raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);
```

Preconditions

- The source code files for devices and any other components with the *runTest* methods are available..
- The Domain Profiles files for the Operating Environment are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Determine the devices and any other components that implement the *TestableObject* or *Resource* interface and verify that the *runTest* operation exists. (OE0081)

Note: Devices may be determined by using the results of test procedure, OE_TC_119.

1. **Pass**: The component implements the *runTest* operation.
2. **Fail**: The component does not implement the *runTest* operation.
3. **N/A**: The *runTest* operation is an empty implementation.

For each component that supports the *runTest* operation, perform the following steps:

B. Verify that the *runTest* operation of the component returns the result values of each test in the testValues parameter. (OE0081)

1. **Pass**: The *runTest* operation returns the results of each test in the testValues parameter.
2. **Fail**: The *runTest* operation does not return the result values of each test in the testValues parameter.

Manual Test Steps

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

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

OE_TC_043				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Determine the devices and any other components that implement the *TestableObject* or *Resource* interface and verify that the *runTest* operation exists. (OE0081) **Note: Devices may be determined by using the results of test procedure, OE_TC_119.**				
1. Examine the Core Framework documentation, and/or source code files and Domain Profile files to determine the devices.	**N/A:** There are no devices in the Operating Environment. (OE0081)			
2. Locate the source code files for the devices.	**N/A:** The source code files of the devices are not located. (OE0081)			
3. Examine the Operating Environment documentation, and/or source code files and Domain Profile files to determine any other component that implements the *Resource* or *TestableObject* interface.	**N/A:** There are no components that support tests in the Operating Environment (OE0081)		In other words, testable objects are those that implement the *TestableObject* interface or the Resource interface. The Resource interface by default inherits the *TestableObject* interface, as well as *PortSupplier*, *PropertySet* and *LifeCycle* interfaces.	

OE_TC_043				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
4. Examine the source code of the components determined from steps 1-3 and verify that the *runTest* operation is implemented for the component. List the source code file of the *runTest* operation for each component.	**Pass**: The *runTest* operation is implemented. (OE0081) **Fail**: The devices or other components do not implement the *runTest* operation. (OE0081) **N/A**: The component has an empty implementation of the *runTest* operation. (OE0081)		Since the component supports the *runTest* operation from Core Framework documentation, source code and/or Domain Profile files, the *runTest* operation should be implemented. A component may inherit the *runTest* functionality from another component or interface. An empty *runTest* operation may support the ability to throw exceptions, this is acceptable.	
For each component that supports the *runTest* operation, perform the following steps:				
B. Verify that the *runTest* operation of the component returns the result values of each test in the testValues parameter. (OE0081)				
5. Examine the source code of the *runTest* operation for each component and verify each test returns the its results in the corresponding value of the id/value pair in the CF::Properties testValues parameter.	**Pass**: The *runTest* operation returns the results of each test in the testValues parameter. (OE0081) **Fail**: The *runTest* operation does not return the result values of each test in the testValues parameter. (OE0081)		Normally a CF::Properties data type is a list of at least one id/value pairs. In this case, the testId would be the id and the result of the test would be inserted into the value option of the id/value pair. The testIds are initially obtained from the data passed into the testValues parameter.	
End of Test				

Test Recording Log – OE_TC_043					
Step 1 (devices names)	Step 2 (source code file of devices)	Step 3 (Other components that implement *TestableObject* or Resource interface?)	Step 4 (*runTest* implemented?) (Pass/Fail/ Untested?)	Step 4 cont'd (list source code file of *runTest* operation)	Step 5 (results of each test returned in the testValues parameter?) (Pass/Fail?)

Test Summary OE_TC_043

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

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

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

OE0081_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.25. OE_TC_045 - TestableObject :: runTest validates testValues parameter

Test Case Number: OE_TC_045

TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0083	All testValues parameter properties (i.e., test properties defined in the propertyfile(s) referenced in the component's SPD) shall be validated.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1.3
SCA Appendix D: Domain Profile	FINAL / 15 May 2006	Page D-4, Section D.2 Page D-8, Section D.2.1.6.1 Page D-19, Section D.4 Page D-24, Section D.4.1.3

Test Objective

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.

Places to Verify

Devices and any other CF Resource components with runTest methods (derived from TestableObject).

IDL References

Data

typedef sequence <DataType> Properties;

Exceptions

exception UnknownTest {
};
exception UnknownProperties { CF::Properties invalidProperties;

};

Operations

void runTest (in unsigned long testId, inout CF::Properties testValues)
raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);

Preconditions

- The source code files are available.
- The SPD and PRF XML file associated with each component is available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate the source code that implements the *runTest()* operation's Built-In Test. (OE0083).
   1. **Pass**: Can locate the code.
   2. **N/A:** Cannot locate the code.

B. For each component, analyze the *runTest()* code to determine that the testValues parameter properties are validated (OE0083).
   1. **Pass:** The testValues parameter properties are validated.
   2. **Fail:** The testValues parameter properties are not validated.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_045				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the source code that implements the *runTest* operation's Built-in Test (OE0083).				
1. Locate the source code that define a Built-in Test (BIT) in the *runTest()* operation.	**Pass**: Can locate the code. (OE0083) **N/A**: Cannot locate the code. (OE0083)		*runTest()* is related to TestableObject (N/A: There is no SCA requirement to have a Built-in-Test.)	
B. For each component, analyze the *runTest()* code to determine that the testValues parameter properties are validated (OE0083).				
2. Verify that the input testValues parameter is validated before being used to provide additional information to run a test.	**Pass**: The testValues parameter properties are validated. (OE0083) **Fail**: The testValues parameter properties are not validated. (OE0083)		The input parameter testValues contain any CF DataTypes that are not known by the component's test implementation or any values that are out of range for the requested test. The testValues properties id(s) that are not known by the component or the value(s) are out of range.	
End of Test				

Test Recording Log OE_TC_045		
Step 1 (*runTest()*)	Step 2 (validated testValues parameter properties) (Yes / No)	Notes

Test Summary OE_TC_045

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

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

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

OE0083_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.26. OE_TC_046 - TestableObject :: runTest validates parameters

Test Case Number: OE_TC_046

TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0084	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.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-9 thru D-25, Section D.4

Test Objective

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.

Places to Verify

Components that use *runTest* methods (derived from TestableObject).

IDL References

Data

CF::Properties

Exceptions

CF::TestableObject::UnknownTest,
CF::UnknownProperties;

Operations

void runTest (in unsigned long testId, inout CF::Properties testValues)

raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);

Preconditions

- The Testable Objects (e.g., components) source code files are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate all components that implement the *runTest* operation (OE0084).
   1. **Pass**: Can locate components.
   2. **N/A**: No components implement the *runTest* operation.

B. For each component identified in Step A, review the component source code. (OE0084)
   1. Locate in the *runTest* operation source code the testId and its testValues
      a. **Pass:** Can locate the testId/testValues.
      b. **Fail:** Cannot locate the testId/testValues.
   2. Verify that when the testValues' values do not match the properties expected by the specified test, then the *runTest* testing is not executed.
      a. Verify that the *runTest* operation's testing is not executed when the testId input, which identifies the test to be run, is not known by the component.
         i. **Pass:** Testing is not executed when the input testId is unknown.
         ii. **Fail:** Testing is executed when the input testId is unknown.
      b. Verify that the *runTest* operation's testing is not executed when the properties provided in the input testValues are out of range.
         i. **Pass**: Testing is not executed when the properties provided in the input testValues are out of range.
         ii. **Fail:** Testing is executed when the properties provided in the input testValues are out of range.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_046				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify all components that implement the *runTest* operation. (OE0084)				
1. Locate all components that implement the *runTest* operation.	**Pass**: Can locate components. (OE0084) **N/A**: Cannot locate components that implement the *runTest* operation. If the *runTest* operation is not implemented by the developer, then end the test case. (OE0084)			
B. For each component identified in Step A, review the component source code.				
B.1 Locate the testId/testValues of the test properties in the source code. (OE0084)				
2. Locate the testId/testValues in the source code. (OE0084)	**Pass**: Can locate the testId/testValues. (OE0084) **Fail**: Cannot locate the testId/testValues. (OE0084)		SCA: 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 property file which is referenced in the component's SPD. These test properties are referred to as the testValues parameter properties	
B.2. Verify that when the testValues' values do not match the properties expected by the specified test, then the *runTest* testing is not executed. (OE0084)				

OE_TC_046				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the *runTest* operation's testing is not executed when the testId input, which identifies the test to be run, is not known by the component.	**Pass**: Testing is not executed when the input testId is unknown. (OE0084) **Fail**: Testing is executed when the input testId is unknown. (OE0084)		Example of a *runTest()* unsigned long testId input: *void runTest(CORBA::ULong testId, CF::Properties& testValues CF_ENV_ARG_DECL);*	
4. Verify that the *runTest* operation's testing is not executed when the properties provided in the input testValues are out of range.	**Pass**: Testing is not executed when the properties provided in the input testValues are out of range. (OE0084) **Fail**: Testing is executed when the properties provided in the input testValues are out of range.(OE0084)			
End of Test				

Test Recording Log – OE_TC_046			
Step 1 (Component)	Step 2 (Id/Values)	Step 3 (known testId?) (Yes / No)	Step 4 (testValues within valid range) (Yes / No)

Test Summary OE_TC_046

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

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

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

OE0084_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.27. OE_TC_047 - PropertySet :: configure

Test Case Number: OE_TC_047
PropertySet::configure

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0091	The *configure* operation shall assign values to the properties as indicated in the input configProperties parameter.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Pages 3-14, Section 3.1.3.1.5.5.1.3
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-15 – C-16, Section C.1

Test Objective

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.

Places to Verify

DomainManager, DeviceManager, Devices, or any other component that inherits the *PropertySet* interface and contains properties.

IDL References

Data:

typedef sequence <DataType> Properties
struct DataType {string id; any value;}

Exceptions:

exception InvalidConfiguration { string msg; CF::Properties invalidProperties;};
exception PartialConfiguration { CF::Properties invalidProperties; };

Operations:

void *configure* (in CF::Properties configProperties)

raises (CF::PropertySet::InvalidConfiguration, CF::PropertySet::PartialConfiguration);

Preconditions

- The source code of the Core Framework (CF) components to verify is available.
- The IDL for the *PropertySet* interface, which includes the *configure* operation, has been validated against the SCA. (OE0703)
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Verify that all the components of the Core Framework which implement the *PropertySet* interface also implement the *configure* operation. (OE0091)
   1. **Pass**: The components of the Core Framework implement the *configure* operation.
   2. **Fail**: There are no implementations of the *configure* operation in the CF.

For each implementation of the *configure* operation in the Core Framework components, perform the following steps:

B. Verify that the *configure* operation assigns the value associated with the id from the *configProperties* parameter to the corresponding value of the property with the same id in the component. (OE0091)
   1. **Pass**: The *configure* operation assigns the value associated with the id of the *configProperties* parameter to the value of the property with the same id in the component.
   2. **Fail**: The *configure* operation does not assign the value associated with the id of the *configProperties* parameter to the value of the property with the same id in the component.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_047				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Verify that all the components of the Core Framework which implement the PropertySet interface also implement the configure operation. (OE0091)				
1. Identify all implementations of the *configure* operation in the source code of the Core Framework. List the components that implement the *configure* operation and list the source code files of the *configure* operation.	**Pass:** The components of the Core Framework implement the *configure* operation. (OE0091) **Fail:** There are no implementations of the *configure* operation in the CF. (OE0091)		Components and their associated source code files that extend(inherit) from the *PropertySet* interface should either contain or inherit the *configure* operation.	
For each implementation of the *configure operation* in the Core Framework components, perform the following steps:				
B. Verify that the *configure* operation assigns the value associated with the id from the *configProperties* parameter to the value of the property with the same id in the component and the value is stored. (OE0091)				
2. Verify in the source code of the component that the id of the *configProperties* parameter is used to determine the property of the CF that is being configured.	**Pass:** The id of the *configProperties* parameter is used to determine the property of the CF that is being configured to enable the property being assigned and stored. (OE0091) **Fail:** The id of the *configProperties* parameter is not used to determine the property of the CF that is being configured; Therefore, the property cannot be assigned and stored. (OE0091)			

OE_TC_047				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the *configure* operation assigns the value associated with the id from the *configProperties* parameter to the corresponding value of the property with the same id in the component.	**Pass:** The *configure* operation assigns the value associated with the id of the *configProperties* parameter to the value of the property with the same id in the component. (OE0091) **Fail:** The *configure* operation does not assign the value associated with the id of the *configProperties* parameter to the value of the property with the same id in the component (OE0091)			
End of Test				

Test Recording Log – OE_TC_047			
Step1a (list the components implementing the *configure* operation)	**Step1b** (list source code file(s) of the *configure* operation)	**Step2** (Is id of *configProperties* parameter used to determine CF property being configured?) (Pass/Fail?)	**Step3** (Is value of *configProperty* assigned to value of the property with the same id in component being configured?) (Pass/Fail?)

Test Summary OE_TC_047

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

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

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

OE0091_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.28. OE_TC_048 - PropertySet :: configure property minimums

Test Case Number: OE_TC_048
PropertySet::configure

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0092	Valid properties for the *configure* operation shall at a minimum be the *configure* readwrite and writeonly properties referenced in the component's SPD.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-14, Section 3.1.3.1.5.5.1.3
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-16; Section C-1
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Pages D-4 – D-5; Section D.2 Pages D-19 to D-24, Section D.4

Test Objective

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.

Places to Verify

Any component of the Core Framework that implements the *PropertySet* or the *Resource* interface(i.e. configurable), such as devices, services, Device Managers, and the Domain Manager.

IDL References

Data

typedef sequence <DataType> Properties
struct DataType {string id; any value;}

Exceptions

CF::PropertySet::InvalidConfiguration
CF::PropertySet::PartialConfiguration

Operations

void *configure* (in CF::Properties configProperties)
raises (CF::PropertySet::InvalidConfiguration, CF::PropertySet::PartialConfiguration);

Preconditions

- The source code files of the components of Core Framework(CF) are available.
- All of the Domain Profiles of the CF are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Identify the readwrite and/or writeonly properties of the Core Framework components by examining the property files referenced in the SPD file of each of the configurable components. (OE0092)
   1. **N/A**: The SPD file of a Core Framework component does not contain a reference to readwrite and/or writeonly properties.
   2. **Fail**: The SPD file of one or more of the Core Framework components is not available.

For each component of the Core Framework that contains configurable properties referenced in its SPD, perform the following steps:

B. Verify that the *configure* operation is implemented for the component. (OE0092)
   1. **Fail**: The configure operation is not implemented for the component.

C. Verify that the *configure* operation of the component supports all the readwrite and/or writeonly properties referenced in its SPD file. (OE0092)
   1. **Pass**: The *configure* operation of the component supports all the readwrite and/or writeonly properties referenced in its SPD file.
   2. **Fail**: The *configure* operation of the component does not support one or more of the readwrite and/or writeonly properties referenced in its SPD file.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_048				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the readwrite and/or writeonly properties of the Core Framework components by examining the property files referenced in the SPD file of each of the configurable components. (OE0092)				
1. Open DCD file(s) and record the SPD files for each of the configurable components.	**Fail**: The SPD of one or more of the Core Framework components is not available. (OE0092)		A declaration for an SPD file in the DCD file may look similar to this. <componentfile id="DCE:xxx-yyy-xxx ="SPD"> <localfile name="../xxxxx.spd.xml"/> </componentfile>	
2. Examine each SPD file from step 1 and identify the names of the PRF files that are referenced within the softpkg element of the SPD file. If the SPD file references SCD files, examine those SCD files for references to additional PRF files. List all of the PRF files.	**N/A**: The Core Framework component does not have a reference to PRF files, therefore the SPD file cannot reference properties. (OE0092)		A PRF reference in an SPD may look similar to this. <propertyfile> <localfile name="zzzzz.prf.xml"/> </propertyfile> An SCD reference in an SPD file may look similar to this. descriptor> <localfile name="xxxxxx.scd.xml"/> </descriptor>	
3. Examine all the PRF files found in step 2 and identify the readwrite or writeonly properties where the simple type element contains a mode element equaling "readwrite" or "writeonly" and where the kindtype element equals "configure". List the	**N/A:** The PRF file(s) referenced by the SPD files of the Core Framework component does not contain readwrite or writeonly properties. **(**OE0092)		The properties referenced in the PRF file of the component's SPD have precedence over the properties referenced by the PRF files from the SCD files.	

OE_TC_048				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
ids of the readwrite and writeonly properties for each component.			A simple id property, with a mode attribute may look similar to this: <simple id="xxx_ID" type="short" mode="readwrite" name="xxxxx "> <value>5</value> <range max="50" min="0"/>	
For each component of the Core Framework that contains configurable properties referenced in its SPD, perform the following steps:				
B. Verify that the *configure* operation is implemented for the component. (OE0092)				
4. Verify that the configure operation is either implemented or inherited for component corresponding to each SPD file that has configurable properties. List the name of the source code file that contain the configure operation corresponding to the component.	**Fail:** The *configure* operation is not implemented for one or more of the configurable components of the Core Framework. (OE0092)		This step can be done by performing a search in the source code for all *configure* operations	
C. Verify that the *configure* operation of the components supports all the readwrite and/or writeonly properties referenced in its SPD file. (OE0092)				

OE_TC_048				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
5. Verify in the source code of the component's *configure* operation that the component contains logic to support the read/write and/or write-only properties found its PRF files referenced from SPD and/or SCD files.	**Pass**: The *configure* operation contains logic to support the read/write and/or write-only properties found in the PRF files referenced from the SPD and SCD files. (OE0092) **Fail**: The *configure* operation does not contain logic to support the read/write and/or write-only properties found in its PRF files referenced from the SPD and/or SCD files. (OE0092)		A CF component may have a bigger set of properties than the set of configurable properties listed in its PRF. However, its *configure* operation must properly handle the set of configurable properties found in the PRF.	
End of Test				

Test Recording Log – OE_TC_048					
Step 1 (DCD file Name)	Step 2 (SPD file Name)	Step 3 (PRF file Name)	Step 4 (configurable properties)	Step 5 (Configure operation file name)	Step 6 (property id supported in the configure operation?) (Pass/Fail?)

Test Summary OE_TC_048

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

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

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

OE0092_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.29. OE_TC_049 - PropertySet :: configure raises PartialConfiguration

Test Case Number: OE_TC_049

PropertySet::configure raises PartialConfiguration exception

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0093	The *configure* operation shall raise a *PartialConfiguration* exception when some configuration properties were successfully set and some configuration properties were not successfully set.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-13, Section 3.1.3.1.5.3.2

Test Objective

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.

Places to Verify

Devices, DeviceManagers, and DomainManagers and other components with configure methods such as services (derived from PropertySet).

IDL References

Exceptions

exception PartialConfiguration { CF::Properties invalidProperties; };

Operation

void configure (in CF::Properties configProperties)
raises (CF::PropertySet::InvalidConfiguration, CF::PropertySet::PartialConfiguration);

Preconditions

- The OE source code files having the *configure* operation are available.

- All the Domain Profile files are available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

For each component having the *configure* operation, perform the following steps:

A. Identify the implementation of the *configure* operation in the source code. (OE0093)
   1. **Pass:** The implementation of the *configure* operation is found.
   2. **Fail:** The implementation of the *configure* operation is not found.

B. Verify that the *PartialConfiguration* exception will be raised when only some of the configuration properties are successfully set. (OE0093)
   1. **Pass:** The *PartialConfiguration* exception is raised where only some of the configuration properties are successfully set.
   2. **Fail:** The *PartialConfiguration* exception is not raised where only some of the configuration properties are successfully set.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_049				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
For each component having the *configure* operation, perform D and E:				
A. Identify the implementation of the *configure* operation in the source code. (OE0093)				
1. Identify and record the source code file(s) containing the *configure* operation.	**Pass:** One or more implementation of the *configure* operation is found. (OE0093) **Fail:** An implementation of the *configure* operation is not found**. (OE0093)**			
B. Verify that the *PartialConfiguration* exception will be raised when only some configuration properties are successfully set. (OE0093)				
2. Verify that the *configure* operation uses the input parameter *configProperties* as type *CF::Properties*.			Example may look like this: void DeviceManager_Servant::configure(const CF::Properties::configProperties)	

OE_TC_049				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the *PartialConfiguration* exception is raised where some configuration properties were successfully set and some configuration properties were not successfully set.	**Pass:** The *PartialConfiguration* exception is raised where some configuration properties were successfully set and some configuration properties were not successfully set. (OE0093) **Fail:** The *PartialConfiguration* exception is not raised where some configuration properties were successfully set and some configuration properties were not successfully set. **(**OE0093)			
End of Test				

Test Recording Log – OE_TC_049			
Step 1 (*configure* operation source code)	**Step 2** (*configure* operation uses input parameters)	**Step 3** (*PartialConfiguration* raised – Y/N?)	**Notes**

Test Summary OE_TC_049

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

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

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

OE0093_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.30. OE_TC_050 - Resource :: start raises StartError

Test Case Number: OE_TC_050

CF::Resource::*start* raises *StartError*

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0099	The errorNumber parameter shall indicate a CF ErrorNumberType value.
OE0105	The *start* operation shall raise the StartError exception if an error occurs while starting the resource.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-16, Section 3.1.3.1.6.5.1.5 Page 3-16, Section 3.1.3.1.6.3.1 Page 3-94, Section 3.1.3.6.13
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-23. C-24

Test Objective

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.

Places to Verify

Source Code Files with the *Resource start* operation

IDL References

Data

enum ErrorNumberType {

CF_NOTSET, CF_E2BIG, CF_EACCES, CF_EAGAIN, CF_EBADF, CF_EBADMSG, CF_EBUSY, CF_ECANCELED, CF_ECHILD, CF_EDEADLK, CF_EDOM, CF_EEXIST, CF_EFAULT, CF_EFBIG, CF_EINPROGRESS, CF_EINTR, CF_EINVAL, CF_EIO, CF_EISDIR, CF_EMFILE, CF_EMLINK, CF_EMSGSIZE, CF_ENAMETOOLONG, CF_ENFILE, CF_ENODEV, CF_ENOENT, CF_ENOEXEC, CF_ENOLCK, CF_ENOMEM, CF_ENOSPC, CF_ENOSYS, CF_ENOTDIR,

CF_ENOTEMPTY, CF_ENOTSUP, CF_ENOTTY, CF_ENXIO, CF_EPERM, CF_EPIPE, CF_ERANGE, CF_EROFS, CF_ESPIPE, CF_ESRCH, CF_ETIMEDOUT, CF_EXDEV};

Exceptions

StartError { CF::ErrorNumberType errorNumber; string msg; };

Operations

void start () raises (CF::Resource::StartError);

Preconditions

- The source code files for components implementing the *Resource* interface are available.
- OE_TC_119 will provide a list of resource components for testing.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate the source code for the *Resource start* operation for the component. (OE0099, OE0105)
   1. **Pass**: The source code for the *Resource start* operation is found.
   2. **Untested:** The source code for the *Resource start* operation is not found.

B. Verify that if a start error occurs, the StartError exception will be raised. (OE0105)
   1. **Pass:** The StartError exception is raised for an occurrence of a start error.
   2. **Fail:** The StartError exception is not raised for an occurrence of a start error.

C. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0099)
   1. **Pass:** The StartError exception provides an error number of ErrorNumberType.
   2. **Fail:** The StartError exception does not provide an error number of ErrorNumberType.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_050				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate that the source code for the *Resource start* operation for the component. (OE0099, OE0105)				
1. Locate all instances of the *start* operation and list the file and line number.	**Pass**: Source code for the *start* operation is found. (OE0099, OE0105)			
B. Verify that if a start error occurs, the StartError exception will be raised. (OE0105)				
2. Determine that a StartError exception occurs if the *start* operation detects an error.	**Pass**: A StartError exception occurs when the *start* operation detects an error. (OE0105 **Fail**: A StartError exception does not occur when the *start* operation detects an error. (OE0105)			
C. Verify that the error number that accompanies the exception is a CF ErrorNumberType value. (OE0099)				
3. Determine if the exception includes an error number (ErrorNumber) of the type ErrorNumberType.	**Pass**: An errorNumber of an ErrorNumberType is passed to the exception. (OE0099) **Fail**: An errorNumber of an ErrorNumberType is not passed to the exception. (OE0099)			
End of Test				

Test Recording Log – OE_TC_050		
Step 1 (*start* operation file and line no.)	Step 2 (exception raised)	Step 4 (errorNumber of ErrorNumberType)

Test Summary OE_TC_050

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

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

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

OE0099_____

OE0105_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.31. OE_TC_064 - Resource :: stop raises StopError

Test Case Number: OE_TC_064

Resource::stop raises StopError

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0103	The *stop* operation shall raise the StopError exception if an error occurs while stopping the resource.
OE0100	The errorNumber parameter shall indicate a CF ErrorNumberType value.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Pages 3-15 through 3-17, Section 3.1.3.1.6;
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Pages C-3 through C-4, Section C.1

Test Objective

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.

Places to Verify

Devices, Application, and any other component with the *stop()* operation (derived from CF::Resource interface)

IDL References

Data

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

Exceptions

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

Operations

void *stop ()* raises (CF::Resource::StopError);

Preconditions

- The CF source code files are available.
- OE_TC_119 will provide a list of resource components for testing.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate all implementations of the *stop()* operation for the component.
B. Verify that the StopError exception is raised when appropriate. (OE0103)
    1. **Pass:** Exception is raised when errors are detected.
    2. **Fail**: An exception is raised when no errors were detected.
    3. **Fail**: No exception is raised when errors were detected.
C. Verify that the error number that accompanies the exception is a CF::ErrorNumberType value. (OE0100)
    1. **Pass:** The errorNumber is in the correct range.
    2. **Fail:** The errorNumber is not in the correct range.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_064				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate all implementations of the stop() operation for the component.				
1. Search the CF devices and application source code for all instantiations of the *stop* operation and record the file and line number.	Instantiations of the *stop* operation will be found.		An Integrated Development Environment (IDE) is the best tool for this search.	
B. Verify that the StopError exception is raised when appropriate. (OE0103)				
2. Verify that the *stop* operation detects errors and to throw exceptions.	**Pass**: Exception is raised when errors are detected. (OE0103). **Fail**: An exception is raised when no errors were detected. (OE0103). **Fail**: No exception is raised when errors were detected. (OE0103).		C/C++/Java uses "try/catch' blocks to detect and catch errors. Other languages may use other methods.	
C. Verify that the error number that accompanies the exception is a CF::ErrorNumberType value (OE0100).				
3. Determine if the exception includes an error number (ErrorNumber) of the type ErrorNumberType.	**Pass**: An errorNumber of an ErrorNumberType is passed to the exception. (OE0100) **Fail**: An errorNumber of an ErrorNumberType is not passed to the exception. (OE0100)			
End of Test				

Test Recording Log – OE_TC_064		
Step 1(*Stop* operation – file and line number)	Step 2 (error exceptions raised)	Step 3 (ErrorNumber valid)

Test Summary OE_TC_064

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

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

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

OE0103_____

OE0100_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.32. OE_TC_071 - Resource :: stop

Test Case Number: OE_TC_071

CF::Resource::Stop

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0102	The *stop* operation shall disable all current operations and put the resource in a non-operating condition.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Pages 3-15 through 3-17, Section 3.1.3.1.6;

Test Objective

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.

Places to Verify

Devices and any other components with *stop* operation (derived from Resource).

IDL References

Operations

void stop () raises (CF::Resource::StopError);

Preconditions

- The CF source code files are available.
- OE_TC_119 will provide a list of resource components for testing.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Locate all implementations of the *stop* operation for the component. (OE0102)
   1. **N/A:** There are no implementations of the *stop* operation.

B. Verify that the *stop* operation disables all current operations of the component. (OE0102)
   1. **Pass:** The *stop* operation stops the component.
   2. **Fail:** The *stop* operation does not prevent the component from continuing to operate.

C. Verify that the *stop* operation puts the component in a non-operating condition. (OE0102)
   1. **Pass:** The component is put into a non-operating condition.
   2. **Fail:** The component can continue to operate after the *stop* operation has been invoked.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_071				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate all implementations of the *stop* operation for the component. (OE0102)				
1. Search the source code for all implementations of the *stop* operation and record the file and line number.	The *stop* operation is found.		An Integrated Development Environment (IDE) is the best tool for this search.	
B. Verify that the *stop* operation disables all current operations of the component. (OE0102)				
2. Verify that the *stop* operation disables all current operations.	**Pass**: The *stop* operation ends current processing. (OE0102) **Fail**: The *stop* operation does not stop current processing. (OE0102)		The spirit of the requirement is that if there are some threads running within the component's address space, then these threads should be terminated inside the stop() operation. If there wasn't any threads started during start() or initialize(), then essentially the stop() doesn't have a current operation to stop.	
C. Verify that the *stop* operation puts the resource in a non-operating condition. (OE0102)				
3. Verify that the resource is put into a non-operating condition.	**Pass**: The *stop* operation is put into a non-operating condition. (OE0102) **Fail**: The *stop* operation is not put into a non-operating condition. (OE0102)		The design documents should describe the non-operating conditions of the system.	
End of Test				

Test Recording Log – OE_TC_071		
Step 1 (file and line number)	Step 2 (current processing ends)	Step 3 (non-operating condition)

Test Summary OE_TC_071

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

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

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

OE0102_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.33. OE_TC_072 - TestableObject :: runTest parameters

Test Case Number: OE_TC_072

Resource::TestableObject::runTest

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0082	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).

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Page D-19, Section D.4.1 Page D-24, Section D.4.1.3 Page D-25, Section D.4.1.4

Test Objective

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.

Places to Verify

Devices and any other component such as resource and Services with *runTest* operations (inherited from *TestableObject)*

IDL References

Exceptions

```
exception UnknownTest {
};
exception UnknownProperties { CF::Properties invalidProperties;
```

};

Operations

void runTest (in unsigned long testid, inout CF::Properties testValues
)
raises (CF::TestableObject::UnknownTest, CF::UnknownProperties);
};

Preconditions

- Domain Profile files are available.
- Domain Profile test has passed.
- The OE source code files having the *runTest* operation and *TestableObject* interface are available.
- The OE documentation is available.
- OE_TC_102 (Base Application Interfaces) test has passed.

Test Description

A. Find all occurrences of the Device Configuration Descriptor (DCD) file. (OE0082)
    1. Determine the names of the devices of the Operating Environment.
        a. **Untested**: A device is not found.

For each device, perform the following steps:

B. Find all occurrences of the Device Package Descriptor (DPD) files. (OE0082)
    1. **Untested**: A DPD file does not exist.
        a. Locate the Properties Descriptor files from each DPD file.
            i. **Untested**: A Properties Descriptor file does not exist.

C. Find all occurrences of the Software Package Descriptor (SPD) files. (OE0082)
    1. **Pass:** One or more SPD files exist.
    2. **Fail**: A SPD file does not exist.
        a. Locate the Properties Descriptor files from each SPD file.
            1. **Untested**: A Properties Descriptor file does not exist.

D. Find all occurrences of the Software Component Descriptor (SCD) files from each SPD file. (OE0082)
    1. **Untested:** A SCD file does not exist.
        a. Locate the Properties Descriptor files in the SCD file.
            1. Untested: A Properties Descriptor file does not exist.

For each Properties Descriptor file (PRF) found, perform the following:

E. Locate the *kind* element and verify that the *kindtype* attribute is set to "test". (OE0082)
   1. **Pass:** The *kind* element *kindtype* attribute is set to "test".
   2. **N/A:** The *kind* element *kindtype* attribute is not set to "test".

F. Verify that the *inputvalue* element is declared in the properties *test* element. (OE0082)
   1. **Pass**: The *inputvalue* element is declared.
   2. **N/A:** The *inputvalue* element is not declared.

G. Verify that the *resultvalue* element is declared in the properties *test* element. (OE0082)
   1. **Pass**: The *resultvalue* element is declared.
   2. **Fail:** The *resultvalue* element is not declared.

For each device found in Step A.1, perform the following:

H. Verify that the source code containing the *runTest* operation can be found. (OE0082)
   1. **Pass:** The source code containing the *runTest* operation is found.
   2. **Fail:** The source code containing the *runTest* operation is not found.

I. Verify that the *runTest* operation uses, at the minimum, *testId* and *testValues* defined in the properties *test* element of the component's Properties Descriptor. (OE0082)
   1. **N/A:** There are no tests in the PRF file or there is no PRF file.
   2. **Pass**: The *runTest* operation uses the *testId* and *testValues* defined in the properties *test* element in the component's Properties Descriptor.
   3. **Fail:** The *runTest* operation does not use the *testId* and *testValues* defined in the properties *test* element in the component's Properties Descriptor.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Find all occurrences of the Device Configuration Descriptor (DCD) file. (OE0082)				
A.1 Determine the names of the devices of the Operating Environment.				
1. Locate all DCD files in the OE. Record the names of the devices.	**Untested**: A device is not found. (OE0082)		Each DCD may describe several devices controlled by the DeviceManager. A device's id and name are usually specified in the DCD file as in the following example: <componentplacement> <componentfileref refid="DCE:dddddxxxxx"/> <componentinstantiation id="DCE.ddd123xxxxx"> <usagename>DeviceName</usagename> </componentinstantiation>	
For each device, perform the following steps:				
B. Find all occurrences of the Device Package Descriptor (DPD) files.				

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. Locate all Device Package Descriptor files.	**Untested**: A DPD file does not exist. (OE0082)		<componentplacement> <componentfileref refid="red id"/> <deployondevice refid="SPD-xxx> <devicepkgfile> <localfile name="xxx.dpd.xml"/> </devicepkgfile>	
B.a Locate the Properties Descriptor files from each DPD file. (OE0082)				
3. Locate all Properties Descriptor from each DPD file. Record the PRF file name.	**Untested**: A PRF file does not exist. (OE0082)		<hwdeviceregistration id="DCE:nnn" name="device" version="1"> <propertyfile type=""> <localfile name="xxx.prf.xml"/> </propertyfile>	
C. Find all occurrences of the Software Package Descriptor (SPD) files. (OE0082)				
4. Locate all Software Package Descriptor files.	**Pass:** One or more SPD files exist. (OE0082) **Fail**: A SPD file does not exist. (OE0082)		<deviceconfiguration id="DCE:nnn" name="Device"> <description>text </description> <devicemanagersoftpkg> <localfile name="xxx.spd.xml"/> </devicemanagersoftpkg>	
C.a Locate all Properties Descriptor files from each SPD file. (OE0082)				

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
5. Locate all Properties Descriptor from ***Step 4***. Record the PRF file name.	**Untested**: A PRF file does not exist. (OE0082)		<propertyfile type=""> <localfile name="xxx.prf.xml"/> </propertyfile>	
D. Find all occurrences of the Software Component Descriptor (SCD) file from each SPD file. (OE0082)				
6. Locate all Software Component Descriptor files from ***Step 4***.	**Untested**: A SCD file does not exist. (OE0082)		<descriptor name=""> <localfile name="xxx.scd.xml"/> </descriptor>	
D.a Locate a Properties Descriptor in each SCD file. (OE0082)				
7. Locate all Properties Descriptor from ***Step 6***. Record the PRF name.	**Untested**: A PRF file does not exist. (OE0082)		<propertyfile type=""> <localfile name="xxx.prf.xml"/> </propertyfile>	
For each Properties Descriptor file (PRF) found, perform the following:				
E. Locate the *kind* element and verify that the *kindtype* attribute is set to "test". (OE0082).				
8. Locate the *kind* element in the PRF file. Verify that the *kindtype* attribute is set to "test".	**Pass:** The *kind* element *kindtype* attribute is set to "test". (OE0082) **N/A:** The *kind* element *kindtype* attribute is not set to "test". (OE0082)		<simple id="yyyy" type="ulong" name="Test" mode="readonly"> <description>Test interface</description> <value>1</value> <kind kindtype="test"/> </simple> Note: kindtype "test" is optional. The expected result should be either "Pass" or "N/A"	

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
8a. If *kindtype* is set to "test" then locate all the *test* elements in the prf. For each *test* element found, perform steps 9 through 10.				
F. Verify that the *inputvalue* element is declared in the properties *test* element. (OE0082)				
9. Locate the *test* element and verify that it has the *inputvalue* element to perform a component test. Ensure also that the simple properties the *inputvalue* element contains must have a *kindtype* value of *test*. Record the value associated with ***test id***.	**Pass:** A *test* element has the *inputvalue* element. (OE0082) **N/A:** A *test* element does not have the *inputvalue* element. (OE0082)		<test id="test_ID"> <description>text </description> <inputvalue> <simple id="runtest_1" type="ulong" name="runtest" mode="readonly"> <description>text </description> <value>1</value> <kind kindtype="test"/> </simple> </inputvalue> Note: "*inputvalue*" is optional. The expected result should be either "Pass" or "N/A"	
G. Verify that the *resultvalue* element is declared in the properties *test* element. (OE0082)				

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
10. Locate the *test* element and verify that it has the *resultvalue* element to perform a component test. Ensure also that the simple properties the *resultvalue* element contains must have a *kindtype* value of *test*.	**Pass:** A *test* element has the *resultvalue* element. (OE0082) **Fail:** A *test* element does not have the *resultvalue* element. (OE0082)		<resultvalue> <simple id="return_value" type = "ulong" name = "result"> <value>2</value> </simple> </resultvalue>	
For each device found in Step A.1, perform the following:				
H. Verify that the source code containing the *runTest* operation can be found. (OE0082)				
11. Locate the source code for the device and record the name of the file.	**Pass:** The source code of the devices is located. (OE0082) **Untested:** The source code of the devices is not located. (OE0082)		The development engineer and the developer documentation, if available, is the best source for this information.	
12. Locate all implementations of the Device-related *runTest* operation in the source code.	**Pass**: The source code containing the *runTest* operation is found. (OE0082) **Fail**: The source code containing the *runTest* operation is not found. (OE0082)		void CF_Resource_i::runTest(CORBA::ULong testId, CF::Properties& testValues CF_ENV_ARG_DECL) {	
I. Verify that the *runTest* operation uses, at the minimum, *testId* and *testValues* defined in the properties *test* element of the component's Properties Descriptor. (OE0082)				

OE_TC_072				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
13. Verify that the *runTest* operation uses *testId* and *testValues* as the test properties.	**N/A:** There are no tests in the PRF file or there is no PRF file. **Pass**: The *runTest* operation uses the *testId* and *testValues* as test properties as defined in the properties *test* element in the component's Properties Descriptor. (OE0082) **Fail:** The *runTest* operation does not use the *testId* and *testValues* as test properties as defined in the properties *test* element in the component's Properties Descriptor. (OE0082)		void CF_Resource_i::runTest(CORBA::ULong testId, CF::Properties& testValues CF_ENV_ARG_DECL) The *runTest*() is allowed to use additional *testId* and *testValues* other than those from the PRFs. However, if the PRFs define any testId and testValues, they should all be used/considered in the runTest operation.	
End of Test				

Test Recording Log – OE_TC_072			
Step1 (device names)			
Step3 (PRF files from DPD)			
Step5 (PRF files from SPD)			
Step7 (PRF files from SCD)			
Step9 (test id & *inputvalue* – Y/N?)			
Step10 (*resultvalue* – Y/N?)			
Step11 (source code file)			
Step13 (testId and testValues – Y/N?)			

Test Summary OE_TC_072

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

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

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

OE0082_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.34. OE_TC_093 - TestableObject :: runTest exception parameter

Test Case Number: OE_TC_093

TestableObject::runTest Exceptions/Errors

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0087	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

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 3-11, Section 3.1.3.1.3.5.1.5;
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	Page C-15
SCA Appendix D: Domain Profile	Version 2.2.2 15 May 2006	Page D-22, Section D.4.1.1.6 paragraph 2 Page D-24, Section D.4.1.3

Test Objective

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.

Places to Verify

Devices

IDL References

Data

```
struct DataType { string id;
                  any value; };
typedef sequence <DataType> Properties;
```

Exceptions

exception *UnknownProperties* { CF::Properties invalidProperties; };

Operations

void *runTest*(in unsigned long testId, inout Properties testValues) raises (UnknownTest, UnknownProperties);

Preconditions

- The *runTest* operation implementation source code files are available.

Test Description

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

A. Identify the source code files and record the file names that implement the *runTest* operation. (OE0087)
   1. **Pass:** The *runTest* operation implementation is located for the device.
   2. **Fail:** The *runTest* operation implementation is not located for the device.

B. Verify that the *runTest* operation supports built-in tests. (OE0087)
   1. **Pass:** The *runTest* operation implementation supports built-in tests.
   2. **N/A:** The *runTest* operation implementation does not support any built-in tests and only throws the CF::TestableObject::UnknownTest exception.

Note: As stated in the requirement invalid testValues are value(s) that are not known by the component or the value(s) are out of range.

C. Verify that all invalid testValues are listed by their id(s) in the *UnknownProperties* exception invalidProperties parameter when the *UnknownProperties* exception is raised by the *runTest* operation when an invalid testValue parameter is received. (OE0087)
   1. **Pass:** The *UnknownProperties* exception parameter contains all the invalid testValues id(s).
   2. **Fail:** The *UnknownProperties* exception parameter does not contain all of the invalid testValues id(s).

Manual Test Steps

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

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

OE_TC_093				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
For each device within the OE under test, perform the following steps:				
A. Identify the source code files that implement the *runTest* operation. (OE0087)				
1. Identify the source code file(s) and record the source file(s) names where the *runTest* operation is implemented for the device.	The *runTest* operation is implemented for the device		void runTest(in unsigned long testId, inout Properties testValues) raises (UnknownTest, UnknownProperties);	
B. Verify that the *runTest* operation supports built-in tests. (OE0087)				
2. Verify that the *runTest* operation supports built-in tests.	**Pass:** The runTest operation supports built-in tests. **N/A:** The runTest operation does not support any built-in tests, and only throws the UnknownTest exception.		If the test only raises the UnknownTest exception OE0087 is Not Applicable.	
C. Verify that all invalid testValues are contained in the *UnknownProperties* exception invalidProperties parameter when the *UnknownProperties* exception is raised by the *runTest* operation when an invalid testValue parameter is received. (OE0087)				

OE_TC_093				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that all invalid testValues id(s) are listed in the *UnknownProperties* exception invalidProperties parameter when the *UnknownProperties* exception is raised when an invalid testValue parameter is received by the *runTest* operation.	**Pass:** All of the invalid testValues id(s) are listed in the invalidProperties parameter when the *UnknownProperties* exception is raised. (**OE0087**) **Fail:** Not all of the invalid testValues id(s) are listed in the invalidProperties parameter when the *UnknownProperties* exception is raised. (**OE0087**)		All of the invalid properties id must be sent in the *UnknownProperties* exception.	
End of Test				

Test Recording Log – OE_TC_093	
Step 1 (source file name with *runTest* implementation)	**Step 2** (all testValues id(s) listed runTest raises the *UnknownProperties* exception?)

Test Summary OE_TC_093

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

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

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

OE0087_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

B.1.35. OE_TC_102 - Base Application Interfaces

Test Case Number: OE_TC_102

CF Interfaces

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
OE0703	Base Application Interfaces shall be implemented using the CF IDL presented in Appendix C.

References

Document Name	Version/Date	Location (Pages, Section)
Software Communications Architecture (SCA)	Version 2.2.2 15 May 2006	Page 2-8, Section 2.2.4.6, Page 3-6, Section 3.1.3.1 Page 3-95, Section 3.2.1.3
SCA Appendix C: Core Framework IDL	Version 2.2.2 15 May 2006	All

Test Objective

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.

Places to Verify

OE IDL files

IDL References

Exceptions

```
exception StartError { CF::ErrorNumberType errorNumber; string msg; };
exception StopError { CF::ErrorNumberType errorNumber; string msg; };
exception InitializeError { CF::StringSequence errorMessages; };
exception ReleaseError { CF::StringSequence errorMessages; };
exception UnknownTest { };
exception InvalidConfiguration {string msg; CF::Properties invalidProperties; };
exception PartialConfiguration { CF::Properties invalidProperties; };
exception UnknownFileSystemProperties { CF::Properties invalidProperties; };
```

```
exception UnknownPort { };
exception InvalidPort { unsigned short errorCode; string msg; };
exception OccupiedPort { };
exception InvalidPort { unsigned short errorCode; string msg; };
exception CreateResourceFailure {CF::ErrorNumberType errorNumber; string msg; };
exception InvalidResourceId { };
exception ShutdownFailure {string msg; };
```

Operations

```
interface Resource {
        void start () raises (CF::Resource::StartError);
        void stop () raises (CF::Resource::StopError);
};
interface LifeCycle {
        void initialize  () raises (CF::LifeCycle::InitializeError);
        void releaseObject () raises (CF::LifeCycle::ReleaseError);
};
interface PropertySet {
        void configure ( in CF::Properties configProperties ) raises
                (CF::PropertySet::InvalidConfiguration, CF::PropertySet::PartialConfiguration);
        void query ( inout CF::Properties configProperties ) raises (CF::UnknownProperties);
};
interface PortSupplier {
        Object getPort ( in string name) raises (CF::PortSupplier::UnknownPort);
};
interface Port {
        void connectPort ( in Object connection, in string connectionId ) raises (CF::Port::InvalidPort, CF::Port::OccupiedPort);
        void disconnectPort ( in string connectionId ) raises (CF::Port::InvalidPort);
};
interface TestableObject {
        void runTest ( in unsigned long testid, inout CF::Properties testValues ) raises
                (CF::TestableObject::UnknownTest, CF::UnknownProperties);
};
```

```
interface ResourceFactory {
        CF::Resource createResource ( in string resourceId, in CF::Properties qualifiers ) raises
                (CF::ResourceFactory::CreateResourceFailure);
        void releaseResource ( in string resourceId ) raises (CF::ResourceFactory::InvalidResourceId);
        void shutdown ( ) raises (CF::ResourceFactory::ShutdownFailure); };
```

Preconditions

- The OE Core Framework IDL files are available.
- The SCA v2.2.2 Appendix C – Core Framework IDL is available

Test Description

A. Locate the OE CF IDL files. (OE0703)
   1. **Pass:** The OE CF IDL files are found.
   2. **Fail:** The OE CF IDL files are not found.

B. Identify the Base Application Interfaces in the OE IDL files. (OE0703)
   Mandatory interfaces are Resource, LifeCycle, PropertySet, PortSupplier, and Port.
   Optional interfaces are TestableObject and ResourceFactory.

   1. **Pass:** The mandatory interfaces are found in the OE IDL files.
   2. **Fail:** A mandatory interface is not found in the OE IDL files.

For each Base Application Interface identified, perform the following:

C. Verify that the OE CF IDL matches the SCA CF IDL, except for white space and comment differences. (OE0703)
   1. **Pass**: The OE CF IDL matches the SCA CF IDL.
   2. **Fail**: The OE CF IDL does not match the SCA CF IDL.

Manual Test Steps

Notes: 1. Test Result will include Pass, Fail, Untested, or N/A.
2. The Test Recording Log is intended to record data for each step that requires recording of data.

OE_TC_102				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Locate the OE CF IDL files. (OE0703)				
1. Identify the OE IDL files for the Core Framework.	**Pass:** The OE CF IDL files are found. (OE0703) **Fail:** The OE CF IDL files are not found. (OE0703)		Consult the developer or review the developer documentation to locate the Core Framework IDL files.	
B. Identify the Base Application Interfaces in the OE IDL files. (OE0703)				
2. Search for all Base Application Interfaces, both mandatory and optional interfaces, in the OE IDL files. Record the interface name.	**Pass:** The mandatory interfaces are found in the OE IDL files. (OE0703) **Fail:** A mandatory interface is not found in the OE IDL files. (OE0703)		**Mandatory interfaces:** • Resource • LifeCycle • PropertySet • PortSupplier • Port **Optional interfaces:** • TestableObject • ResourceFactory	
For each Base Application Interface identified from Step 2, perform the following step:				
C. Verify that the OE CF IDL matches the SCA CF IDL, except for white space and comment differences. (OE0703)				

OE_TC_102				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
3. Verify that the OE CF IDL matches the SCA CF IDL seen in SCA v2.2.2 Appendix C.	**Pass**: The OE CF IDL matches the SCA CF IDL.. (OE0703) **Fail**: The OE CF IDL does not match the SCA CF IDL.. (OE0703)			
End of Test				

Test Recording Log – OE_TC_102		
Step 1 (OE IDL files)	**Step 2** (Interface name)	**Step 3** (OE IDL matches SCA IDL **Y/N?**)

Test Summary OE_TC_102

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

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

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

OE0703_____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer:_____

Date Tested: _____

Witness: _____

Index of Test Case Titles for Manual Tests

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
AEP Applications have no abnormal termination	5	31	B.5.4.	OE_TC_021
AEP file mode creation masks	5	37	B.5.5.	OE_TC_044
AEP mandatory functions	5	142	B.5.18.	OE_TC_131
Aggregate Device	3	191	B.3.25.	OE_TC_096
AggregateDevice devices	3	12	B.3.2.	OE_TC_022
AggregateDevice :: addDevice	3	20	B.3.3.	OE_TC_023
AggregateDevice :: addDevice FAILURE_ALARM	3	29	B.3.4.	OE_TC_025
AggregateDevice :: addDevice raises InvalidObjectReference	3	35	B.3.5.	OE_TC_027
AggregateDevice :: removeDevice	3	42	B.3.6.	OE_TC_028
AggregateDevice :: removeDevice FAILURE_ALARM	3	48	B.3.7.	OE_TC_029
AggregateDevice :: removeDevice raises InvalidObjectReference	3	54	B.3.8.	OE_TC_030
Application :: releaseObject releases all objects	2	96	B.2.16.	OE_TC_108
Application Attributes	2	258	B.2.39.	OE_TC_233
Application Delegates Implementation of Resource operations	2	237	B.2.38.	OE_TC_189
ApplicationFactory :: create	2	84	B.2.14.	OE_TC_074
ApplicationFactory :: create deallocates capacity on devices	2	201	B.2.32.	OE_TC_162
ApplicationFactory :: create defines an order for initialization	2	212	B.2.34.	OE_TC_165
ApplicationFactory :: create does property comparisons	2	119	B.2.20.	OE_TC_121
ApplicationFactory :: create establishes connections for named applications	2	207	B.2.33.	OE_TC_164
ApplicationFactory :: create raises CreateApplicationError	2	138	B.2.23.	OE_TC_125
ApplicationFactory Attributes	2	258	B.2.41.	OE_TC_236
Base Application Interfaces	1	263	B.1.35.	OE_TC_102
Base Device Interfaces	3	71	B.3.10.	OE_TC_062
Component Identifier's execute parameter	5	200	B.5.22.	OE_TC_161
CORBA :: CosEventComm	5	52	B.5.8.	OE_TC_063
CORBA :: Log Producer	5	15	B.5.2.	OE_TC_002

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
CORBA :: NamingService	5	5	B.5.1.	OE_TC_001
Device :: adminState attribute changes	3	154	B.3.20.	OE_TC_088
Device :: adminState commanded to be LOCKED	3	91	B.3.12.	OE_TC_080
Device :: allocateCapacity	3	6	B.3.1.	OE_TC_004
Device :: allocateCapacity	3	105	B.3.13.	OE_TC_081
Device :: allocateCapacity BUSY	3	114	B.3.14.	OE_TC_082
Device :: allocateCapacity Failure	3	122	B.3.15.	OE_TC_083
Device :: allocateCapacity raises exceptions	3	299	B.3.39.	OE_TC_229
Device :: allocateCapacity's acceptable properties	3	217	B.3.28.	OE_TC_118
Device :: deallocateCapacity	3	128	B.3.16.	OE_TC_084
Device :: deallocateCapacity raises InvalidCapacity	3	141	B.3.18.	OE_TC_086
Device :: deallocateCapacity raises InvalidState	3	305	B.3.40.	OE_TC_230
Device :: deallocateCapacity usageState	3	134	B.3.17.	OE_TC_085
Device :: Logical Device - CORBA	3	177	B.3.23.	OE_TC_094
Device :: Logical Device - CORBA Register	3	184	B.3.24.	OE_TC_095
Device :: Logical Device allocation properties	3	206	B.3.27.	OE_TC_098
Device :: Logical Device executable parameters	3	168	B.3.22.	OE_TC_092
Device :: Logical Devices - CF Interfaces	3	198	B.3.26.	OE_TC_097
Device :: releaseObject	3	147	B.3.19.	OE_TC_087
Device :: releaseObject raises ReleaseError exception	3	162	B.3.21.	OE_TC_089
Device :: usageState	3	77	B.3.11.	OE_TC_079
Device Attributes	3	261	B.3.35.	OE_TC_218
Device operationalState	3	60	B.3.9.	OE_TC_058
DeviceManager :: getComponentImplementationId	2	89	B.2.15.	OE_TC_090
DeviceManager Attributes	2	247	B.2.40.	OE_TC_224
DeviceManager Register	2	17	B.2.3.	OE_TC_033
DeviceManager Register and the DCD file	2	23	B.2.4.	OE_TC_034
DeviceManager startup process	2	277	B.2.42.	OE_TC_285

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
DeviceManager's execparamproperties	2	101	B.2.17.	OE_TC_112
Domain Manager logs defined in DMD file	2	49	B.2.8.	OE_TC_054
Domain Manager services defined in DMD file	2	61	B.2.10.	OE_TC_056
Domain Profile	5	58	B.5.9.	OE_TC_070
Domain Profile files	5	114	B.5.17.	OE_TC_119
DomainManager :: installApplication raises ApplicationAlreadyInstalled	2	11	B.2.2.	OE_TC_026
DomainManager :: installApplication raises ApplicationInstallationError	2	67	B.2.11.	OE_TC_057
DomainManager :: installApplication raises InvalidFileName	2	172	B.2.28.	OE_TC_132
DomainManager :: registerDevice	2	107	B.2.18.	OE_TC_113
DomainManager :: registerDevice raises RegisterError	2	132	B.2.22.	OE_TC_124
DomainManager :: registerDevice registers if device doesn't exist	2	230	B.2.37.	OE_TC_168
DomainManager :: registerDevice returns without error if device exists	2	224	B.2.36.	OE_TC_167
DomainManager :: registerDevice verifies input parameters	2	218	B.2.35.	OE_TC_166
DomainManager :: registerDeviceManager	2	6	B.2.1.	OE_TC_024
DomainManager :: registerDeviceManager establishes connections	2	188	B.2.30.	OE_TC_157
DomainManager :: registerDeviceManager raises RegisterError	2	146	B.2.24.	OE_TC_126
DomainManager :: registerDeviceManager sends event to ODM	2	180	B.2.29.	OE_TC_154
DomainManager :: registerService	2	73	B.2.12.	OE_TC_065
DomainManager :: registerService	2	79	B.2.13.	OE_TC_073
DomainManager :: registerService raises RegisterError	2	153	B.2.25.	OE_TC_127
DomainManager :: uninstallApplication raises ApplicationUninstallationError	2	125	B.2.21.	OE_TC_122
DomainManager :: unregisterDevice raises UnregisterError	2	166	B.2.27.	OE_TC_129
DomainManager :: unregisterDeviceManager	2	43	B.2.7.	OE_TC_053
DomainManager :: unregisterDeviceManager	2	194	B.2.31.	OE_TC_158
DomainManager :: unregisterDeviceManager raises UnregisterError	2	159	B.2.26.	OE_TC_128
DomainManager :: unregisterService	2	32	B.2.5.	OE_TC_051
DomainManager :: unregisterService client-side	2	38	B.2.6.	OE_TC_052
DomainManager :: unregisterService raises UnregisterError	2	55	B.2.9.	OE_TC_055

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
DTD files	5	108	B.5.16.	OE_TC_117
Event Service	5	23	B.5.3.	OE_TC_003
Exceptions from the Mounted FileSystem	4	18	B.4.2.	OE_TC_040
ExecutableDevice :: execute	3	290	B.3.38.	OE_TC_227
ExecutableDevice :: execute raises exceptions	3	278	B.3.37.	OE_TC_222
ExecutableDevice :: execute raises InvalidFileName	3	244	B.3.32.	OE_TC_135
ExecutableDevice :: terminate raises InvalidProcess	3	250	B.3.33.	OE_TC_145
ExecutableDevice Types	3	272	B.3.36.	OE_TC_221
File :: close	4	55	B.4.5.	OE_TC_067
File :: close raises FileException	4	60	B.4.6.	OE_TC_068
File :: read raises IOException	4	49	B.4.4.	OE_TC_066
File :: setFilePointer raises FileException	4	66	B.4.7.	OE_TC_069
File :: sizeOf raises FileException	4	215	B.4.29.	OE_TC_147
File :: write raises IOException	4	208	B.4.28.	OE_TC_146
File Attributes	4	262	B.4.36.	OE_TC_265
FileManager :: list	4	6	B.4.1.	OE_TC_031
FileManager :: mount	4	278	B.4.38.	OE_TC_276
FileManager :: mount raises InvalidFileName	4	194	B.4.26.	OE_TC_143
FileSystem :: copy	4	99	B.4.12.	OE_TC_106
FileSystem :: copy raises InvalidFileName when inputs are invalid	4	148	B.4.20.	OE_TC_137
FileSystem :: copy raises InvalidFileName when inputs are the same	4	202	B.4.27.	OE_TC_144
FileSystem :: create	4	106	B.4.13.	OE_TC_107
FileSystem :: create raises FileException	4	227	B.4.31.	OE_TC_149
FileSystem :: create raises InvalidFileName	4	165	B.4.22.	OE_TC_139
FileSystem :: exists	4	159	B.4.21.	OE_TC_138
FileSystem :: exists raises InvalidFileName	4	254	B.4.35.	OE_TC_159
FileSystem :: list raises FileException	4	87	B.4.10.	OE_TC_104
FileSystem :: list raises InvalidFileName	4	78	B.4.9.	OE_TC_103

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
FileSystem:: mkdir	4	129	B.4.17.	OE_TC_114
FileSystem:: mkdir raises FileException	4	241	B.4.33.	OE_TC_151
FileSystem:: mkdir raises InvalidFileName	4	179	B.4.24.	OE_TC_141
FileSystem:: open raises FileException	4	233	B.4.32.	OE_TC_150
FileSystem:: open raises InvalidFileName	4	172	B.4.23.	OE_TC_140
FileSystem:: query Input Parameter	4	269	B.4.37.	OE_TC_275
FileSystem:: remove raises FileException	4	93	B.4.11.	OE_TC_105
FileSystem:: remove raises InvalidFileName	4	141	B.4.19.	OE_TC_136
FileSystem:: rmdir	4	135	B.4.18.	OE_TC_115
FileSystem:: rmdir raises FileException	4	247	B.4.34.	OE_TC_152
FileSystem:: rmdir raises InvalidFileName	4	187	B.4.25.	OE_TC_142
FileSystem create Responsibilities	4	286	B.4.39.	OE_TC_283
FileSystem Filename Lengths	4	38	B.4.3.	OE_TC_060
FileSystem::copy raises FileException	4	221	B.4.30.	OE_TC_148
FileSystem's CREATED_TIME_ID property	4	111	B.4.14.	OE_TC_109
FileSystem's LAST_ACCESS_TIME_ID property	4	123	B.4.16.	OE_TC_111
FileSystem's MODIFIED_TIME_ID property	4	117	B.4.15.	OE_TC_110
Framework Control Interfaces	2	112	B.2.19.	OE_TC_116
Framework Services Interfaces	4	72	B.4.8.	OE_TC_091
General Rules : Higher Order Language	5	90	B.5.13.	OE_TC_099
Hardware Critical Interfaces	5	103	B.5.15.	OE_TC_101
Legacy Software interfaces	5	95	B.5.14.	OE_TC_100
LifeCycle :: initialize raises InitializeError	1	168	B.1.19.	OE_TC_037
LifeCycle :: releaseObject	1	173	B.1.20.	OE_TC_038
LifeCycle :: releaseObject raises ReleaseError	1	179	B.1.21.	OE_TC_039
LoadableDevice :: load	3	256	B.3.34.	OE_TC_153
LoadableDevice :: load raises InvalidFileName	3	230	B.3.30.	OE_TC_133
LoadableDevice :: load raises LoadFail	3	224	B.3.29.	OE_TC_130

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
LoadableDevice :: unload raises InvalidFileName	3	237	B.3.31.	OE_TC_134
Mandatory interfaces of the AEP	5	43	B.5.6.	OE_TC_059
Minimum CORBA	5	47	B.5.7.	OE_TC_061
Name Binding's execute parameter	5	194	B.5.21.	OE_TC_160
Naming Context creation	5	186	B.5.20.	OE_TC_156
Naming Context's execute parameter	5	180	B.5.19.	OE_TC_155
Networking AEP	5	208	B.5.23.	OE_TC_290
OMG Lightweight Log Service :: administrative_state	1	37	B.1.5.	OE_TC_009
OMG Lightweight Log Service :: clear_log	1	131	B.1.14.	OE_TC_018
OMG Lightweight Log Service :: destroy	1	139	B.1.15.	OE_TC_019
OMG Lightweight Log Service :: get_availability_status	1	31	B.1.4.	OE_TC_008
OMG Lightweight Log Service :: get_n_records	1	20	B.1.2.	OE_TC_006
OMG Lightweight Log Service :: get_operational_state	1	45	B.1.6.	OE_TC_010
OMG Lightweight Log Service :: get_record_id_from_time	1	51	B.1.7.	OE_TC_011
OMG Lightweight Log Service :: LogFullAction	1	25	B.1.3.	OE_TC_007
OMG Lightweight Log Service :: LogStatus	1	8	B.1.1.	OE_TC_005
OMG Lightweight Log Service :: retrieve_records	1	56	B.1.8.	OE_TC_012
OMG Lightweight Log Service :: retrieve_records_by_level	1	67	B.1.9.	OE_TC_013
OMG Lightweight Log Service :: retrieve_records_by_producer_id	1	78	B.1.10.	OE_TC_014
OMG Lightweight Log Service :: retrieve_records_by_producer_name	1	91	B.1.11.	OE_TC_015
OMG Lightweight Log Service :: write_record	1	117	B.1.13.	OE_TC_017
OMG Lightweight Log Service :: write_records	1	103	B.1.12.	OE_TC_016
Port :: connectPort	1	147	B.1.16.	OE_TC_020
Port :: connectPort raises OccupiedPort	1	155	B.1.17.	OE_TC_032
Port :: disconnectPort	1	161	B.1.18.	OE_TC_035
PropertySet :: configure	1	212	B.1.27.	OE_TC_047
PropertySet :: configure property minimums	1	218	B.1.28.	OE_TC_048
PropertySet :: configure raises PartialConfiguration	1	225	B.1.29.	OE_TC_049

Test Case Title	App B Volume #	Page	Section Number	Test Case Number
Provided interfaces described as provides ports	5	67	B.5.10.	OE_TC_075
Required interfaces described as uses ports	5	73	B.5.11.	OE_TC_076
Resource :: start raises StartError	1	231	B.1.30.	OE_TC_050
Resource :: stop	1	241	B.1.32.	OE_TC_071
Resource :: stop raises StopError	1	236	B.1.31.	OE_TC_064
SCA APIs and non-IDL interfaces	5	79	B.5.12.	OE_TC_077
TestableObject :: runTest exception parameter	1	257	B.1.34.	OE_TC_093
TestableObject :: runTest parameters	1	246	B.1.33.	OE_TC_072
TestableObject :: runTest returns results	1	195	B.1.24.	OE_TC_043
TestableObject :: runTest uses testId parameter	1	184	B.1.22.	OE_TC_041
TestableObject :: runTest uses testValues parameter	1	189	B.1.23.	OE_TC_042
TestableObject :: runTest validates parameters	1	206	B.1.26.	OE_TC_046
TestableObject :: runTest validates testValues parameter	1	201	B.1.25.	OE_TC_045
End of Table				