Joint Tactical Networking Center Test and Evaluation Laboratory

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

Appendix C
Application Environment Profile Amended Requirements Test Procedures

13 April 2022

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

Prepared by:
Joint Tactical Networking Center Test and Evaluation Laboratory

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited (13 April 2022). JTNC 2022-1011

TABLE OF CONTENTS

APPENDIX C AEP AMENDED REQUIREMENTS TEST CASES

This Appendix contains the test case procedures for the Application Environment Profile (AEP) Amended requirements. The test case procedures include the test case name, test objective, preconditions, parameters, test description, the name of the related automated test name (if relevant), and the test requirement(s) being tested.

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

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

The definitions for the various test results are as follows:

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

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

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

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

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

C.1 OE_TC_170 - AEP Amended Applications have no abnormal termination

Test Case Number: OE_TC_170

Application abnormal termination

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
OE0798	An application that conforms to the AEP shall not result in abnormal termination of the process because this profile does not support multiple processes.

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B: Application Environment Profile (AEP) Amended	Version 2.2.2A 22 October 2008	Page 5, Section B.4.1.4
IEEE Standard for Information Technology – Standardized Application Environment Profile (AEP) – POSIX® Realtime and Embedded Application Support, IEEE Std 1003.13-2003.	10 September 2004	Page 43, Section 6.3.1

Test Objective

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

Places to Verify

DomainManager, DeviceManager, and devices

IDL References

None

Preconditions

- The completion of testing requirement OE0001.

- The OE under test uses the AEP Amended requirements (versus the original AEP requirements).

Test Description

A. Review the test result for OE0001. (OE0798)
   1. **Pass:** If requirement, OE0001 passed.
   2. **Fail:** If requirement, OE0001 failed.

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_170				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Review the test result for OE0001. (OE0798)				
1. Review and document the test result for OE0001.	**Pass:** OE0001 passed. (OE0798) **Untested**: OE0001 failed. (OE0798)		The passing of this requirement depends on the passing of the SCA requirement, OE0001. This requirement passes if the POSIX implementation is POSIX compliance.	
End of Test				

Test Recording Log OE_TC_170	
Step 1 (test result of OE0001)	

Test Summary OE_TC_170

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):

OE0798 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

C.2 OE_TC_172 - AEP Amended mandatory functions

Test Case Number: OE_TC_172
OS::Application Environment Profile (AEP)

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
OE0790	The function listed in Table B-24 shall behave as described in the referenced clause.
OE0794	The functions listed in Table B-3 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0795	The functions listed in Table B-4 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0796	The functions listed in Table B-5 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0797	The functions listed in Table B-6 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0811	The functions listed in Table B-7 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0800	The functions listed in Table B-8 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0801	The functions listed in Table B-9 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0802	The functions listed in Table B-10 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0804	The functions listed in Table B-11 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0805	The functions listed in Table B-12 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0806	The functions listed in Table B-13 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0807	The functions listed in Table B-14 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0808	The functions listed in Table B-15 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0809	The functions listed in Table B-16 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0810	The functions listed in Table B-17 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.

SCA v2.2.2 Tag	SCA v2.2.2A Text
OE0812	The functions listed in Table B-18 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0813	The functions listed in Table B-19 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0814	The functions listed in Table B-20 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0815	The functions listed in Table B-21 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0816	The functions listed in Table B-22 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0818	The functions listed in Table B-23 shall behave as described in the applicable clauses of the referenced POSIX specifications contained in Table B-1.
OE0817	The options, limits, and any other constraints on POSIX.1 shall be provided as described in Table B-2

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B: SCA Application Environment Profile (AEP) Amended	Version 2.2.2A 22 October 2008	Pages 3 through 18 Sections B.4.1.1 through B.4.1.21
IEEE Standard for Information Technology – Standardized Application Environment Profile (AEP) – POSIX® Realtime and Embedded Application Support, IEEE Std 1003.13-2003.	10 September 2004	Pages 111 thru 121, Annex B
The Open Group Base Specifications Issue 6, IEEE Std 1003.1	2004 Edition	http://www.opengroup.org/onlinepubs/000095399/

Test Objective

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

Places to Verify

OE header files including aio.h, ctype.h, dirent.h, fcntl.h, locale.h, math.h, mqueue.h, pthread.h, sched.h, semaphore.h, setjmp.h, signal.h, stdio.h, stdlib.h, string.h, sys/mman.h, sys/stat.h, sys/types.h, time.h, unistd.h, and utime.h.

IDL References

None.

Preconditions

- The OE header source code files are available.
- The OE under test uses the AEP Amended requirements (versus the original AEP requirements).

Test Description

A. Identify the source code files that implement POSIX interfaces. (OE0790, OE0794, OE0795, OE0796, OE0797, OE0811, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0812, OE0813, OE0814, OE0815, OE0816, OE0818, OE0817)
   1. **N/A:** The source code files are not available.

B. Verify that the options listed as mandatory in table B.2 of Appendix B are defined**.** (OE0817)
   1. **Pass:** All mandatory options are defined and provided for within the context of the OE.
   2. **Fail:** One or more mandatory options are not defined and not provided for within the context of the OE.

C. Verify that the mandatory functions are defined. (OE0790, OE0794, OE0795, OE0796, OE0797, OE0811, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0812, OE0813, OE0814, OE0815, OE0816, OE0818)
   1. **Pass:** All mandatory functions listed in Table B-3 through Table B-24 are defined and used within the context of the OE.
   2. **Fail:** One or more mandatory functions listed in Table B-3 through Table B-24 are not used within the context of the OE.

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_172				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement POSIX interfaces. (OE0790, OE0794, OE0795, OE0796, OE0797, OE0811, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0812, OE0813, OE0814, OE0815, OE0816, OE0818, OE0817)				
1. Perform a search for POSIX interfaces in the OE source code provided by the developer. Record the source file name(s).	**N/A**: The OE does not implement POSIX interfaces. **Pass:** Source code files are found. **Pass:** The developer is using a third party's certified POSIX interface. (OE0790, OE0794, OE0795, OE0796, OE0797, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0811, OE0812, OE0813, OE0814, OE0815, OE0816, OE0817, OE0818)			
For the following steps use the source code found above or the third party's detailed implementation.				
B. Verify that the options listed as mandatory in table B.2 of Appendix B are defined. (OE0817)				
2. Verify that the options (see Table B-2) are provided within the context of the OE. The rows below, from here to Step 3, are the mandatory options (i.e _POSIX_ASYNCHRONOUS_IO is a mandatory option). The comments	**Pass:** All mandatory options are defined and provided for within the context of the OE. (OE0817) **Fail:** One or more mandatory option not provided within the context of the OE. (OE0817)		**Step 2** is in several sections to separate the function signatures for easier reading. More information for each function can be found on this website: http://www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap02.html#tag_02_01_06	

column discusses what must be present for that option to be supported.				

_POSIX_ASYNCHRONOUS_IO
For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.
			#include <aio.h>
			
			int aio_cancel(int *fildes*, struct aiocb ***aiocbp*);
			
			int aio_error(const struct aiocb ***aiocbp*);
			
			int aio_fsync(int *op*, struct aiocb ***aiocbp*
			
			int aio_read(struct aiocb ***aiocbp*);
			
			ssize_t aio_return(struct aiocb ***aiocbp*);
			
			int aio_suspend(const struct aiocb * const *list*[], int *nent*, const struct timespec ***timeout*);
			
			int aio_write(struct aiocb ***aiocbp*);
			
			int lio_listio(int *mode*, struct aiocb *restrict const *list*[restrict], int *nent*, struct sigevent *restrict *sig*);	

_POSIX_MEMLOCK_RANGE For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <sys/mman.h> int **mlock**(const void * *addr*, size_t *len*); int **munlock**(const void * *addr*, size_t *len*);	
_POSIX_MEMLOCK For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <sys/mman.h> int **mlockall**(int *flags*); int **munlockall**(void);	
_POSIX_MESSAGE_PASSING For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <mqueue.h> int **mq_close**(mqd_t *mqdes*);	

			int **mq_getattr**(mqd_t *mqdes*, struct mq_attr **mqstat*); int **mq_notify**(mqd_t *mqdes*, const struct sigevent **notification*); mqd_t **mq_open**(const char **name*, int *oflag*, ...); ssize_t **mq_receive**(mqd_t *mqdes*, char **msg_ptr*, size_t *msg_len*, unsigned int **msg_prio*); int **mq_send**(mqd_t *mqdes*, const char **msg_ptr*, size_t *msg_len*, unsigned int *msg_prio*); int **mq_setattr**(mqd_t *mqdes*, const struct mq_attr **mqstat*, struct mq_attr **omqstat*); int **mq_unlink**(const char **name*);	
_POSIX_REALTIME_SIGNALS For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <sys/types.h> #include <signal.h> int **sigqueue**(pid_t *pid*, int *signo*, const union sigval *value*); int **sigwaitinfo**(const sigset_t **set*, siginfo_t **info*); int **sigtimedwait**(const sigset_t **set*, siginfo_t **info*const struct timespec **timeout*);	
_POSIX_SEMAPHORES For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <semaphore.h> int **sem_close**(sem_t **sem*); int **sem_destroy**(sem_t **sem*);	

			int **sem_getvalue**(sem_t *_sem_, int *_sval_); int **sem_init**(sem_t *_sem_, int _pshared_, unsigned int _value_); sem_t *__sem_open__(const char*_name_, int _oflag_, ...); int **sem_post**(sem_t *_sem_); int **sem_trywait**(sem_t *_sem_); int **sem_wait**(sem_t *_sem_); int **sem_unlink**(const char *_name_);	
_POSIX_THREAD_ATTR_STACKADDR For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <pthread.h> int **pthread_attr_getstackaddr**(const pthread_attr_t *restrict _attr_, void **restrict _stackaddr_); int **pthread_attr_setstackaddr**(pthread_attr_t *_attr_, void *_stackaddr_);	
_POSIX_THREAD_ATTR_STACKSIZE For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <pthread.h> int **pthread_attr_getstack**(const pthread_attr_t *restrict _attr_, void **restrict _stackaddr_, size_t *restrict _stacksize_); int **pthread_attr_setstack**(pthread_attr_t *_attr_, void *_stackaddr_, size_t _stacksize_);	
_POSIX_THREAD_PRIO_INHERIT For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <pthread.h> int **pthread_mutexattr_getprotocol**(const pthread_mutexattr_t *restrict _attr_, int *restrict _protocol_);	

			int **pthread_mutexattr_setprotocol**(pthread_mutexattr_t*_attr_, int _protocol_);	
_POSIX_THREAD_PRIO_PROTECT For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <pthread.h> int **pthread_mutex_getprioceiling**(const pthread_mutex_t *restrict _mutex_, int *restrict _prioceiling_); int **pthread_mutex_setprioceiling**(pthread_mutex_t *restrict _mutex_, int _prioceiling_, int *restrict _old_ceiling_); int **pthread_mutexattr_getprioceiling**(const pthread_mutexattr_t *restrict _attr_, int *restrict _prioceiling_); int **pthread_mutexattr_setprioceiling**(pthread_mutexattr_t*_attr_, int _prioceiling_); int **pthread_mutexattr_getprotocol**(const pthread_mutexattr_t *restrict _attr_, int *restrict _protocol_); int **pthread_mutexattr_setprotocol**(pthread_mutexattr_t*_attr_, int _protocol_);	
_POSIX_THREAD_PRIORITY_SCHEDULING For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <pthread.h> int **pthread_attr_getinheritsched**(const pthread_attr_t *restrict _attr_, int *restrict _inheritsched_); int **pthread_attr_setinheritsched**(pthread_attr_t *_attr_, int _inheritsched_);	

			int **pthread_attr_getscope**(const pthread_attr_t *restrict *attr*, int *restrict *contentionscope*); int **pthread_attr_setscope**(pthread_attr_t **attr*, int *contentionscope*); int **pthread_attr_setschedpolicy**(pthread_attr_t **attr*, int *policy*); int **pthread_attr_getschedpolicy**(const pthread_attr_t **attr*, int **policy*); int **pthread_getschedparam**(pthread_t *thread*, int **policy*, struct sched_param **param*); int **pthread_setschedparam**(pthread_t *thread*, int *policy*, const struct sched_param **param*); int **pthread_setschedprio**(pthread_t *thread*, int *prio*); #include <sched.h> int **sched_get_priority_max**(int *policy*); int **sched_get_priority_min**(int *policy*); int **sched_rr_get_interval**(pid_t *pid*, struct timespec **interval*);	
_POSIX_TIMERS For the above option to be supported by the OE, the utilities, functions and facilities in the Comments column must be present.			#include <time.h> int clock_**getres**(clockid_t *clock_id*, struct timespec **res*); int clock_gettime(clockid_t *clock_id*, struct timespec **tp*);	

			int clock_settime(clockid_t *clock_id*, const struct timespec ***tp*); int **nanosleep**(const struct timespec ***rqtp*, struct timespec ***rmtp*); #include <signal.h> int timer_**create**(clockid_t *clockid*, struct sigevent ***evp*, timer_t ***timerid*); int timer_**delete**(timer_t *timerid*); int timer_**getoverrun**(timer_t *timerid*); int timer_**gettime**(timer_t *timerid*, struct itimerspec ***value*); int timer_**settime**(timer_t *timerid*, int *flags*, const struct itimerspec *restrict *value*, struct itimerspec *restrict *ovalue*);	
C. Verify that the mandatory functions are provided. (OE0790, OE0794, OE0795, OE0796, OE0797, OE0800, OE0801, OE0802, OE0804, OE0805, OE0806, OE0807, OE0808, OE0809, OE0810, OE0811, OE0812, OE0813, OE0814, OE0815, OE0816, OE0818)				
3. Verify that the mandatory functions from table B-3 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0794) **Fail**: The mandatory functions are not provided. (OE0794)		There are no mandatory functions listed in table B-3. All OE's pass this requirement.	
4. Verify that the mandatory functions from table B-4 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0795) **Fail**: The mandatory functions are not provided. (OE0795)		There are no mandatory functions listed in table B-4. All OE's pass this requirement.	
5. Verify that the mandatory functions from table B-5 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0796) **Fail**: The mandatory functions are not provided. (OE0796)		There are no mandatory functions listed in table B-5. All OE's pass this requirement.	

6. Verify that the mandatory functions from table B-6 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0797) **Fail**: The mandatory functions are not provided. (OE0797)		**POSIX_SIGNALS** #include <stdlib.h> void **abort**(void); #include <unistd.h> int **pause**(void); #include <signal.h> int **raise**(int *sig*); int **kill** (*pid_t pid, int signum*); int **sigaction**(int *sig*, const struct sigaction *restrict *act*, struct sigaction *restrict *oact*); int **sigaddset**(sigset_t **set*, int *signo*); int **sigdelset**(sigset_t **set*, int *signo*); int **sigemptyset**(sigset_t **set*); int **sigfillset**(sigset_t **set*); int **sigismember**(const sigset_t **set*, int *signo*); void (***signal**(int *sig*, void (**func*)(int)))(int); int **sigpending**(sigset_t **set*); int **sigprocmask**(int *how*, const sigset_t *restrict *set*, sigset_t *restrict *oset*); int pthread_sigmask(int *how*, const sigset_t *restrict *set*, sigset_t *restrict *oset*); int **sigsuspend**(const sigset_t **sigmask*); int **sigwait**(const sigset_t *restrict *set*, int *restrict *sig*);	

7. Verify that the mandatory functions from table B-7 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0811) **Fail**: The mandatory functions are not provided. (OE0811)		There are no mandatory functions listed in table B-7. All OE's pass this requirement.	
8. Verify that the mandatory functions from table B-8 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0800) **Fail**: The mandatory functions are not provided. (OE0800)		There are no mandatory functions listed in table B-8. All OE's pass this requirement.	
9. Verify that the mandatory functions from table B-9 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0801) **Fail**: The mandatory functions are not provided. (OE0801)		**POSIX_FILE_SYSTEM** #include <fcntl.h> int **creat**(const char *_path_, mode_t _mode_); #include <sys/stat.h> int **mkdir**(const char *_path_, mode_t _mode_); int **fstat**(int _fildes_, struct stat *_buf_); #include <unistd.h> int **access**(const char *_path_, int _amode_); int **chdir**(const char *_path_); long **fpathconf**(int _fildes_, int _name_); long **pathconf**(const char *_path_, int _name_); char *__getcwd__(char *_buf_, size_t _size_); int **link**(const char *_path1_, const char *_path2_); int **unlink**(const char *_path_);	

			#include <dirent.h>
			int **closedir**(DIR *_dirp_);
			
			DIR *__opendir__(const char *_dirname_);struct dirent *__readdir__(DIR *_dirp_);
			
			int __readdir_r__(DIR *restrict _dirp_, struct dirent *restrict _entry_, struct dirent **restrict _result_);
			
			void __rewinddir__(DIR *_dirp_);
			
			#include <stdio.h>
			int __remove__(const char*_path_);
			
			int __rename__(const char *_old_, const char *_new_);
			
			#include <sys/stat.h>
			int __stat__(const char *restrict _path_, struct stat *restrict _buf_);
			
			#include <utime.h>
			int __utime__(const char *_path_, const struct utimbuf*_times_);	

10. Verify that the mandatory functions from table B-10 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0802) **Fail**: The mandatory functions are not provided. (OE0802)		There are no mandatory functions listed in table B-10. All OE's pass this requirement.	

11. Verify that the mandatory functions from table B-11 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0804)
	
	Fail: The mandatory functions are not provided. (OE0804)
			POSIX_FD_MGMT
			#include <stdio.h>
			int __fseek__(FILE *_stream_, long _offset_, int _whence_);
			
			int __fseeko__(FILE *_stream_, off_t _offset_, int _whence_);
			
			void __rewind__(FILE *_stream_);	

			long **ftell**(FILE *_stream_); off_t **ftello**(FILE *_stream_); #include <unistd.h> off_t **lseek**(int _fildes_, off_t _offset_, int _whence_);	

12. Verify that the mandatory functions from table B-12 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0805)
	
	Fail: The mandatory functions are not provided. (OE0805)
			POSIX_DEVICE_IO
			#include <fcntl.h>
			int **open**(const char *_path_, int _oflag_, ...);
			
			#include <stdio.h>
			int **getc**(FILE *_stream_);
			
			int **getchar**(void);
			
			char *__gets__(char *s);
			
			int **fgetc**(FILE *_stream_);
			
			char *__fgets__(char *restrict _s_, int _n_, FILE *restrict _stream_);
			
			int **fileno**(FILE *_stream_);
			
			FILE *__fopen__(const char *restrict _filename_, const char *restrict _mode_);
			
			int **fprintf**(FILE *restrict _stream_, const char *restrict _format_, ...);
			
			int **fputc**(int _c_, FILE *_stream_);
			
			int **fputs**(const char *restrict _s_, FILE *restrict _stream_);
			
			size_t **fread**(void *restrict _ptr_, size_t _size_, size_t _nitems_, FILE *restrict _stream_);	

			FILE *__freopen__(const char*restrict
			
			int __fscanf__(FILE *restrict *stream*, const char *restrict *format*, ...);
			
			size_t __fwrite__(const void *restrict *ptr*, size_t *size*, size_t *nitems*, FILE *restrict *stream*);
			
			void __perror__(const char *s);
			
			int __putc__(int *c*,FILE *_stream*);
			
			int __putchar__(int *c*);
			
			int __puts__(const char*s);
			
			int __printf__(const char *restrict *format*, ...);
			
			int __snprintf__(char *restrict *s*, size_t *n*, const char *restrict *format*, ...);
			
			int __sprintf__(char *restrict *s*, const char *restrict *format*, ...);
			
			int __scanf__(const char *restrict *format*, ...);
			
			int __sscanf__(const char *restrict *s*, const char *restrict *format*, ...);
			
			#include <unistd.h>
			int __close__(int *fildes*);
			
			void __clearerr__(FILE *_stream*);
			
			int __fclose__(FILE *_stream*);
			
			FILE *__fdopen__(int *fildes*, const char*_mode*);	

			int **feof**(FILE *_stream_); int **ferror**(FILE *_stream_); int **fflush**(FILE *_stream_); ssize_t **read**(int _fildes_, void *_buf_, size_t _nbyte_); void **setbuf**(FILE *restrict _stream_, char *restrict _buf_); #include <stdio.h> int **setvbuf**(FILE *restrict _stream_, char *restrict _buf_, int _type_, size_t _size_); int **ungetc**(int _c_, FILE *_stream_); ssize_t **write**(int _fildes_, const void *_buf_, size_t _nbyte_);	
13. Verify that the mandatory functions from table B-13 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0806) **Fail**: The mandatory functions are not provided. (OE0806)		There are no mandatory functions listed in table B-13. All OE's pass this requirement.	
14. Verify that the mandatory functions from table B-14 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0807) **Fail**: The mandatory functions are not provided. (OE0807)		There are no mandatory functions listed in table B-14. All OE's pass this requirement.	
15. Verify that the mandatory functions from table B-15 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0808) **Fail**: The mandatory functions are not provided. (OE0808)		There are no mandatory functions listed in table B-15. All OE's pass this requirement.	
16. Verify that the mandatory functions from table B-16 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0809)		There are no mandatory functions listed in table B-16. All OE's pass this requirement.	

	Fail: The mandatory functions are not provided. (OE0809)			
17. Verify that the mandatory functions from table B-17 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0810) **Fail**: The mandatory functions are not provided. (OE0810)		**POSIX_C_LANG_SUPPORT** #include <stdlib.h> int **abs**(int *i*); double **atof**(const char ***str*); int **atoi**(const char ***str*); long **atol**(const char***str*); void ****bsearch**(const void ***key*, const void ***base*, size_t *nel*, size_t *width*, int (***compar*)(const void *, const void *)); void ****calloc**(size_t *nelem*, size_t *elsize*); void **free**(void ***ptr*); int **fscanf**(FILE *restrict *stream*, const char *restrict *format*, ...); void ****malloc**(size_t *size*); void **qsort**(void ***base*, size_t *nel*, size_t *width*, int (***compar*)(const void *, const void *)); long **labs**(long *i*); long long llabs(long long *i*); int **rand**(void); int **rand_r**(unsigned ***seed*); void ****realloc**(void ***ptr*, size_t *size*); int **scanf**(const char *restrict *format*, ...);	

			int **sscanf**(const char *restrict *s*, const char *restrict *format*, ...);
			
			void **srand**(unsigned *seed*);
			
			double **strtod**(const char *restrict *nptr*, char **restrict *endptr*); float strtof(const char *restrict *nptr*, char **restrict *endptr*); long double strtold(const char *restrict *nptr*, char **restrict *endptr*);
			
			long **strtol**(const char *restrict *str*, char **restrict *endptr*, int *base*); long long strtoll(const char *restrict *str*, char **restrict *endptr*, int *base*)
			
			unsigned long **strtoul**(const char *restrict *str*, char **restrict *endptr*, int *base*); unsigned long long strtoull(const char *restrict *str*, char **restrict *endptr*, int *base*);
			
			#include <time.h>
			
			char ***asctime**(const struct tm **timeptr*); char *asctime_r(const struct tm *restrict tm, char *restrict buf);
			
			char ***ctime**(const time_t **clock*); char ***ctime_r**(const time_t **clock*,char **buf*);
			
			struct tm ***gmtime**(const time_t **timer*); struct tm ***gmtime_r**(const time_t *restrict *timer*, struct tm *restrict *result*); struct tm ***localtime**(const time_t **timer*); struct tm ***localtime_r**(const time_t *restrict *timer*, struct tm *restrict *result*);
			
			time_t **mktime**(struct tm **timeptr*);	

			size_t **strftime**(char *restrict *s*, size_t *maxsize*, const char *restrict *format*, const struct tm *restrict *timeptr*); time_t **time**(time_t **tloc*); #include <ctype.h> int **isalnum**(int *c*); int **isalpha**(int *c*); int **isblank**(int *c*); int **iscntrl**(int *c*); int **isdigit**(int *c*); int **isgraph**(int *c*); int **islower**(int *c*); int **isprint**(int *c*); int **ispunct**(int *c*); int **isspace**(int *c*); int **isupper**(int *c*); int **isxdigit**(int *c*); int **tolower**(int *c*); int **toupper**(int *c*); #include <locale.h> char ***setlocale**(int *category*, const char **locale*);	

			#include <stdio.h> int **fprintf**(FILE *restrict *stream*, const char *restrict *format*, ...); int **printf**(const char *restrict *format*, ...); int **snprintf**(char *restrict *s*, size_t *n*, const char *restrict *format*, ...); int **sprintf**(char *restrict *s*, const char *restrict *format*, ...); #include <string.h> void ***memchr**(const void **s*, int *c*, size_t *n*); int **memcmp**(const void **s1*, const void **s2*, size_t *n*); void ***memcpy**(void *restrict *s1*, const void *restrict *s2*, size_t *n*); void ***memmove**(void **s1*, const void **s2*, size_t *n*); void ***memset**(void *s, int *c*, size_t *n*); char ***strchr**(const char **s*, int *c*); int **strcmp**(const char* *s1*, const char **s2*); int **strcoll**(const char **s1*, const char **s2*); size_t **strcspn**(const char **s1*, const char **s2*); char ***strerror**(int *errnum*);	

			int **strerror_r**(int *errnum*, char* *strerrbuf*, size_t *buflen*); size_t **strlen**(const char *s); char *__restrict__ **strcat**(char *__restrict__ *s1*, const char *__restrict__ *s2*); char *__restrict__ **strncat**(char *__restrict__ *s1*, const char *__restrict__ *s2*, size_t *n*); int **strncmp**(const char *s1, const char *s2, size_t *n*); char *__restrict__ **strcpy**(char *__restrict__ *s1*, const char *__restrict__ *s2*); char *__restrict__ **strncpy**(char *__restrict__ *s1*, const char *__restrict__ *s2*, size_t *n*); char *__restrict__ **strpbrk**(const char *s1, const char *s2); char *__restrict__ **strrchr**(const char *s, int *c*); size_t **strspn**(const char *s1, const char *s2); char *__restrict__ **strstr**(const char *s1, const char *s2); char *__restrict__ **strtok**(char *__restrict__ *s1*, const char *__restrict__ *s2*); char ***strtok_r**(char *__restrict__ *s*, const char *__restrict__ *sep*, char **__restrict__ *lasts*); size_t **strxfrm**(char *__restrict__ *s1*, const char *__restrict__ *s2*, size_t *n*); #include <stdarg.h> *type* **va_arg**(va_list *ap*, *type*); void **va_copy**(va_list *dest*, va_list *src*);	

			void **va_end**(va_list *ap*);
			void **va_start**(va_list *ap*, *argN*);
			
			#include <stdarg.h>
			#include <stdio.h>
			int vfprintf(FILE *restrict *stream*, const char *restrict *format*, va_list *ap*);
			int vprintf(const char *restrict *format*, va_list *ap*);
			
			int **vsnprintf**(char *restrict *s*, size_t *n*, const char *restrict *format*, va_list *ap*);
			int vsprintf(char *restrict *s*, const char *restrict *format*, va_list *ap*);	

18. Verify that the mandatory functions from table B-18 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0812)
	
	Fail: The mandatory functions are not provided. (OE0812)
			POSIX_C_LANG_MATH
			#include <math.h>
			
			double **acos**(double *x*);
			float acosf(float *x*);
			long double acosl(long double *x*);
			
			double **asin**(double *x*);
			float asinf(float *x*);
			long double asinl(long double *x*);
			
			double **atan**(double *x*);
			float atanf(float *x*);
			long double atanl(long double *x*);
			
			double **atan2**(double *y*, double *x*);
			float atan2f(float *y*, float *x*);
			long double atan2l(long double *y*, long double *x*);
			
			double **ceil**(double *x*);
			float ceilf(float *x*);
			long double ceill(long double *x*);
			
			double **cos**(double *x*);
			float cosf(float *x*);	

			long double cosl(long double *x*);
			
			double **cosh**(double *x*)
			float coshf(float *x*);
			long double coshl(long double *x*);
			
			double **exp**(double *x*);
			float expf(float *x*);
			long double expl(long double *x*);
			
			double **fabs**(double *x*);
			float fabsf(float *x*);
			long double fabsl(long double *x*);
			
			double **floor**(double *x*);
			float floorf(float *x*);
			long double floorl(long double *x*);
			
			double **fmod**(double *x*, double *y*);
			float fmodf(float *x*, float *y*);
			long double fmodl(long double *x*, long double *y*);
			
			double **frexp**(double *num*, int **exp*);
			float frexpf(float *num*, int **exp*)
			long double frexpl(long double *num*, int **exp*);
			
			double **ldexp**(double *x*, int *exp*);
			float ldexpf(float *x*, int *exp*);
			long double ldexpl(long double *x*, int *exp*);
			
			double **log**(double *x*);
			float logf(float *x*);
			long double logl(long double *x*);
			
			double **log10**(double *x*);
			float log10f(float *x*);
			long double log10l(long double *x*);
			
			double **modf**(double *x*, double **iptr*);	

			float modff(float *value*, float **iptr*);
			long double modfl(long double *value*, long double **iptr*);
			
			double **pow**(double *x*, double *y*);
			float powf(float *x*, float *y*);
			long double powl(long double *x*, long double *y*);
			
			double **sin**(double *x*);
			float sinf(float *x*);
			long double sinl(long double *x*);
			
			double **sinh**(double *x*);
			float sinhf(float *x*);
			long double sinhl(long double *x*);
			
			double **sqrt**(double *x*);
			float sqrtf(float *x*);
			long double sqrtl(long double *x*);
			
			double **tan**(double *x*);
			float tanf(float *x*);
			long double tanl(long double *x*);
			
			double **tanh**(double *x*);
			float tanhf(float *x*);
			long double tanhl(long double *x*);	

19. Verify that the mandatory functions from table B-19 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0813) **Fail**: The mandatory functions are not provided. (OE0813)		There are no mandatory functions listed in table B-19. All OE's pass this requirement.	
20. Verify that the mandatory functions from table B-20 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0814) **Fail**: The mandatory functions are not provided. (OE0814)		**POSIX_SEMAPHORES** #include <semaphore.h> int **sem_close**(sem_t **sem*); int **sem_destroy**(sem_t **sem*);	

			int **sem_getvalue**(sem_t *_sem_, int *_sval_); int **sem_init**(sem_t *_sem_, int _pshared_, unsigned int _value_); sem_t *__sem_open__(const char *_name_, int _oflag_, ...); int **sem_post**(sem_t *_sem_); int **sem_trywait**(sem_t *_sem_); int **sem_unlink**(const char *_name_); int **sem_wait**(sem_t *_sem_);	
21. Verify that the mandatory functions from table B-21 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0815) **Fail**: The mandatory functions are not provided. (OE0815)		**POSIX_TIMERS** #include <time.h> int **clock_getres**(clockid_t _clock_id_, struct timespec *_res_); int **clock_gettime**(clockid_t _clock_id_, struct timespec *_tp_); int **clock_settime**(clockid_t _clock_id_, const struct timespec *_tp_); int **nanosleep**(const struct timespec *_rqtp_, struct timespec *_rmtp_); #include <signal.h> int **timer_create**(clockid_t _clockid_, struct sigevent *_evp_, timer_t *_timerid_); int **timer_delete**(timer_t _timerid_); int **timer_getoverrun**(timer_t _timerid_); int **timer_gettime**(timer_t _timerid_, struct itimerspec *_value_); int **timer_settime**(timer_t _timerid_, int _flags_,	

			const struct itimerspec *restrict *value*, struct itimerspec *restrict *ovalue*);	
22. Verify that the mandatory functions from table B-22 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0816) **Fail**: The mandatory functions are not provided. (OE0816)		**POSIX_THREADS_BASE** #include <pthread.h> int **pthread_attr_getschedparam**(const pthread_attr_t *restrict *attr*, struct sched_param *restrict *param*); int **pthread_attr_setschedparam**(pthread_attr_t *restrict *attr*, const struct sched_param *restrict *param*); int **pthread_cancel**(pthread_t *thread*); void **pthread_cleanup_pop**(int *execute*); void **pthread_cleanup_push**(void (**routine*)(void*), void **arg*); int **pthread_cond_destroy**(pthread_cond_t **cond*); int **pthread_cond_init**(pthread_cond_t *restrict *cond*, const pthread_condattr_t *restrict *attr*); int pthread_cond_t *cond* = PTHREAD_COND_INITIALIZER; int **pthread_cond_broadcast**(pthread_cond_t **cond*); int **pthread_cond_signal**(pthread_cond_t **cond*); int **pthread_cond_timedwait**(pthread_cond_t *restrict *cond*, pthread_mutex_t *restrict *mutex*, const struct timespec *restrict *abstime*);	

			int **pthread_cond_wait**(pthread_cond_t *restrict *cond*, pthread_mutex_t *restrict *mutex*); int **pthread_condattr_destroy**(pthread_condattr_t **attr*); int **pthread_condattr_init**(pthread_condattr_t **attr*); int **pthread_condattr_getclock**(const pthread_condattr_t *restrict *attr*, clockid_t *restrict *clock_id*); int **pthread_condattr_setclock**(pthread_condattr_t **attr*, clockid_t *clock_id*); int **pthread_condattr_getpshared**(const pthread_condattr_t *restrict *attr*, int *restrict *pshared*); int **pthread_condattr_setpshared**(pthread_condattr_t **attr*, int *pshared*); int **pthread_create**(pthread_t *restrict *thread*, const pthread_attr_t *restrict *attr*, void *(**start_routine*)(void*), void *restrict *arg*); int **pthread_detach**(pthread_t *thread*); int **pthread_equal**(pthread_t *t1*, pthread_t *t2*); void **pthread_exit**(void **value_ptr*);	

			int **pthread_getschedparam**(pthread_t *thread*, int *restrict *policy*, struct sched_param *restrict *param*); int **pthread_setschedparam**(pthread_t *thread*, int *policy*, const struct sched_param **param*); void ***pthread_getspecific**(pthread_key_t *key*); int **pthread_setspecific**(pthread_key_t *key*, const void **value*); int **pthread_join**(pthread_t *thread*, void ***value_ptr*); int **pthread_key_create**(pthread_key_t **key*, void (**destructor*)(void*)); int **pthread_key**_delete(pthread_key_t *key*); int **pthread_kill**(pthread_t *thread*, int *sig*); int **pthread_mutexattr_destroy**(pthread_mutexattr_t **attr*); int **pthread_mutexattr_init**(pthread_mutexattr_t **attr*); int **pthread_mutexattr_getprioceiling**(const pthread_mutexattr_t *restrict *attr*, int *restrict *prioceiling*); int **pthread_mutexattr_setprioceiling**(pthread_mutexattr_t **attr*, int *prioceiling*);	

			int **pthread_mutexattr_getprotocol**(const pthread_mutexattr_t *restrict *attr*, int *restrict *protocol*); int **pthread_mutexattr_setprotocol**(pthread_mutexattr_t **attr*, int *protocol*); int **pthread_mutexattr_getpshared**(const pthread_mutexattr_t *restrict *attr*, int *restrict *pshared*); int **pthread_mutexattr_setpshared**(pthread_mutexattr_t **attr*, int *pshared*); int **pthread_mutexattr_gettype**(const pthread_mutexattr_t *restrict *attr*, int *restrict *type*); int **pthread_mutexattr_settype**(pthread_mutexattr_t **attr*, int *type*); int **pthread_mutex_destroy**(pthread_mutex_t **mutex*); int **pthread_mutex_init**(pthread_mutex_t *restrict *mutex*, const pthread_mutexattr_t *restrict *attr*); pthread_mutex_t *mutex* = PTHREAD_MUTEX_INITIALIZER; int **pthread_mutex_getprioceiling**(const pthread_mutex_t *restrict *mutex*, int *restrict *prioceiling*); int **pthread_mutex_setprioceiling**(pthread_mutex_t *restrict *mutex*, int *prioceiling*, int *restrict *old_ceiling*);	

			int **pthread_mutex_lock**(pthread_mutex_t **mutex*); int **pthread_mutex_trylock**(pthread_mutex_t **mutex*); int **pthread_mutex_unlock**(pthread_mutex_t **mutex*); int **pthread_mutex_timedlock**(pthread_mutex_t **restrict mutex* const struct timespec **restrict abs_timeout*); int **pthread_once**(pthread_once_t **once_control*, void (**init_routine*)(void)); pthread_once_t *once_control* = PTHREAD_ONCE_INIT; pthread_t **pthread_self**(void); int **pthread_setcancelstate**(int *state*, int **oldstate*); int **pthread_setcanceltype**(int *type*, int **oldtype*); void **pthread_testcancel**(void); int **pthread_getschedparam**(pthread_t *thread*, int *restrict policy*, struct sched_param **restrict param*); int **pthread_setschedparam**(pthread_t *thread*, int *policy*, const struct sched_param **param*); void ****pthread_getspecific**(pthread_key_t *key*);	

			int **pthread_setspecific**(pthread_key_t *key*, const void **value*); int **pthread_sigmask**(int *how*, const sigset_t *restrict *set*, sigset_t *restrict *oset*);	

23. Verify that the mandatory functions from table B-23 are provided in the OE header files.	**Pass**: The mandatory functions are provided. (OE0818)
	
	Fail: The mandatory functions are not provided. (OE0818)
			If one or more functions listed below are not provided, then OE0818 fails this step.
			
			POSIX_THREAD_SAFE_FUNCTIONS
			#include <time.h>
			
			char *__asctime__(const struct tm *__timeptr__);
			
			char *__ctime__(const time_t *__clock__);
			
			struct tm *__gmtime__(const time_t *__timer__);
			
			struct tm *__localtime__(const time_t *__timer__);
			struct tm *__localtime_r__(const time_t *restrict __timer__, struct tm *restrict __result__);
			
			#include <stdlib.h>
			
			int rand(void);
			int __rand_r__(unsigned *__seed__);
			
			#include <dirent.h>
			struct dirent *readdir(DIR *__dirp__);
			int __readdir_r__(DIR *restrict __dirp__, struct dirent *restrict __entry__, struct dirent **restrict __result__);
			
			#include <string.h>	

			char *strerror(int *errnum*);
			int **strerror_r**(int *errnum*, char* *strerrbuf*, size_t *buflen*);
			
			char *strtok(char *restrict *s1*, const char *restrict *s2*);
			char ***strtok_r**(char *restrict *s*, const char *restrict *sep*, char **restrict *lasts*);	

24. Verify that the mandatory functions listed in Table B-24 are provided in the OE.	**Pass:** All mandatory functions are provided for within the context of the OE. (OE0790) **Fail:** One or more mandatory functions are not referenced within the context of the OE. (OE0790)		If one or more functions listed below are not provided, then OE0790 fails this step. #include <pthread.h> int **pthread_mutexattr_gettype**(const pthread_mutexattr_t *restrict attr, int *restrict type); int **pthread_mutexattr_settype**(pthread_mutexattr_t *attr, int type);	
End of Test				

Test Recording Log – OE_TC_172	
Step	**Functions not present**
Step 2 (options)	
_POSIX_ASYNCHRONOUS_IO	
_POSIX_MEMLOCK_RANGE	
_POSIX_MEMLOCK	

Test Recording Log – OE_TC_172	
Step	**Functions not present**
_POSIX_MESSAGE_PASSING	
_POSIX_REALTIME_SIGNALS	
_POSIX_SEMAPHORES	
_POSIX_THREAD_ATTR_STACKADDR	
_POSIX_THREAD_ATTR_STACKSIZE	
_POSIX_THREAD_PRIO_INHERIT	
_POSIX_THREAD_PRIO_PROTECT	
_POSIX_THREAD_PRIORITY_SCHEDULING	
_POSIX_TIMERS	
Step 6 (Table B-6)	
POSIX_SIGNALS	
Step 9 (Table B-9)	

Test Recording Log – OE_TC_172	
Step	**Functions not present**
POSIX_FILE_SYSTEM	
Step 11 (Table B-11)	
POSIX_FD_MGMT	
Step 12 (Table B-12)	
POSIX_DEVICE_IO	
Step 17 (Table B-17)	
POSIX_C_LANG_SUPPORT	
Step 18 (Table B-18)	
POSIX_C_LANG_MATH	
Step 20 (Table B-20)	
POSIX_SEMAPHORES	
Step 21 (Table B-21)	
POSIX_TIMERS	
Step 22 (Table B-22)	

Test Recording Log – OE_TC_172	
Step	**Functions not present**
POSIX_THREADS_BASE	
Step 23 (Table B-23)	
POSIX_THREADS_SAFE_FUNCTIONS	
Step 24 (Table B-24)	
XSI_THREAD_MUTEX_EXT	

Test Summary OE_TC_172

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):

OE0790_____

OE0794_____	OE0808_____
OE0795_____	OE0809_____
OE0796_____	OE0810_____
OE0797_____	OE0811_____
OE0800_____	OE0812_____
OE0801_____	OE0813_____
OE0802_____	OE0814_____
OE0804_____	OE0815_____
OE0805_____	OE0816_____
OE0806_____	OE0817_____
OE0807_____	OE0818_____

Failed Items (Section/Step Number):

_____
_____
_____

Test Engineer:_____

Date Tested: _____

Witness: _____

C.3 OE_TC_173 - AEP Amended file mode creation masks

Test Case Number: OE_TC_173
CF::FileSystem

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
OE0803	An application that conforms to the AEP shall be guaranteed that the file mode creation mask for any object created by any process is S-IRWXU; that is, the object shall be fully accessible to the creator.

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B: SCA Application Environment Profile (AEP) Amended	Version 2.2.2.A 22 October 2008	Page 7, Section B.4.1.8
SCA Appendix C: Core Framework IDL	Version 2.2.2 Final / 15 May 2006	Pages C-7, C-8

Test Objective

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

Places to Verify

FileSystem and FileManager

IDL References

Operations

```
void mkdir (in string directoryName )
        raises (CF::InvalidFileName, CF::FileException);

CF::File create ( in string fileName )
        raises (CF::InvalidFileName, CF::FileException);
```

Preconditions

- The FileSystem and FileManager source code files are available.
- The OE under test uses the AEP Amended requirements (versus the original AEP requirements).

Test Description

A. Identify the source code that implements the CF::File *create* operation. (OE0803)
   1. **Fail**: The source code files are not available.
B. Verify that the file mode creation mask for the CF::File *create* operation is at a minimum S-IRWXU. (OE0803)
   1. **Pass:** The file mode creation mask is at a minimum S-IRWXU.
   2. **Fail:** The file mode mask does not include S-IRWXU.
C. Identify the source code that implements the *mkdir* operation. (OE0803)
   1. **Fail**: The source code files are not available.
D. Verify that the directory mode creation mask for the *mkdir* operation is at a minimum S-IRWXU. (OE0803)
   1. **Pass:** The directory mode creation mask is at a minimum S-IRWXU.
   2. **Fail:** The directory mode mask does not include S-IRWXU.

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_173				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code that implements the CF::File *create* operation (OE0803)				
1. Perform a search for the *create* function in the FileManager and FileSystem source code provided by the developer. Record the source file name(s).	**Fail**: The source code files are not available.(OE0803)		The *create* operation provides the ability to create a new plain file on the file system.	
B. Verify that the file mode creation mask for the CF::File *create* operation is at a minimum S-IRWXU. (OE0803))				
2. Verify that the file mode creation mask for the *create* operation will include S-IRWXU.	**Pass**: The file mode creation mask includes S-IRWXU. (OE0803) **Fail**: The file mode creation mask does not include S-IRWXU. (OE0803)		The creation mask allows you to control the file permission bits that are set when creating a file. Typically the mask consists of 3 octal digits, with the first digit being the owner permission, the second being group permission and the third being world (everyone else) permission. Therefore, the octal mask expected would be 7xx where the xx bits do not matter. S - use symbolic permissions R - read W - write	

			X - execute/search U - owner The S-IRWXU is a macro definition for the octal value of 700. The definition is located in sys/stat.h, on most platforms. Code example: "mode_t mode= S_IRWXU|S_IRWXG| S_IRWXO; fileDescriptor = open(filename, mode);" FileManagers will normally delegate the *create* operation to the appropriate FileSystem.	
C. Identify the source code that implements the *mkdir* operation. (OE0803)				
3. Perform a search for the *mkdir* function in the FileManager and FileSystem source code provided by the developer. Record the source file name(s).	**Fail**: The source code files are not available.(OE0803)		The *mkdir* operation provides the ability to create a directory on the file system.	
D. Verify that the directory mode creation mask for the mkdir operation is at a minimum S-IRWXU. (OE0803)				
4. Verify that the directory mode creation mask for the *mkdir* operation will include S-IRWXU.	**Pass**: The directory mode creation mask includes S-IRWXU. (OE0803) **Fail**: The directory mode creation mask does not include S-IRWXU. (OE0803)		The creation mask allows you to control the file permission bits that are set when creating a directory. Typically the mask consists of 3 octal digits, with the first digit being the owner permission, the second	

			being group permission and the third being world (everyone else) permission. Therefore, the octal mask expected would be 7xx where the xx bits do not matter. S - use symbolic permissions R - read W - write X - execute/search U - owner The S-IRWXU is a macro definition for the octal value of 700. The definition is located in sys/stat.h, on most platforms. FileManagers will normally delegate the *mkdir* operation to the appropriate FileSystem.	
End of Test				

Test Recording Log – OE_TC_173			
Step 1 (source files having *create* operation)	**Step 2** (valid mask – Y/N?)	**Step 3** (source files having *mkdir* operation)	**Step 4** (valid mask – Y/N?)

Test Summary OE_TC_173

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):

OE0803 (create) _____

OE0803 (mkdir) _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

C.4 OE_TC_175 - AEP Amended Standard C Library header files

Test Case Number: OE_TC_175

Standard C Library header files

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
OE0791	The Standard C Library header files listed in Table B-25 shall be included within the AEP as described in the referenced clause.

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B: SCA Application Environment Profile (AEP) Amended	Version 2.2.2.A 22 October 2008	Pages 19 through 39, Section B.5
C Standard: Programming Languages – C, ISO/IEC 9899:1999 [C99]	Technical Corrigendum 2 (TC2) May 6, 2005	Page 165, Section 7.1.2

Test Objective

This test case verifies OE0791. The objective of this test is to verify that the Core Framework supports the Standard C Library header files designated as mandatory in Appendix B [Reference 1] of the SCA v2.2.2A.

Places to Verify

Core Framework source code files.

IDL References

None.

Preconditions

- The Core Framework source code files are available.
- The OE under test uses the AEP Amended requirements (versus the original AEP requirements).

Test Description

A. Verify that the mandatory Standard C Library header files listed in Table B-25 of Appendix B [Reference 1] are properly provided. (OE0791)
   1. **Pass:** The mandatory header files are properly defined according to [Reference 1].
   2. **Fail:** The mandatory header files are not properly defined according to [Reference 1].

B. Verify that the mandatory functions listed in each Standard C Library header files, from Step A, are properly provided. (OE0791)
   1. **Pass:** The mandatory functions are provided.
   2. **Fail:** The mandatory functions are not provided.

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.

<table>
<tr><th colspan="5">OE_TC_175</th></tr>
<tr><th>Steps</th><th>Expected Results</th><th>Actual Results</th><th>Comments</th><th>Test Result</th></tr>
<tr><td colspan="5">A. Verify that the mandatory Standard C Library header files listed in Table B-25 of Appendix B [Reference 1] are properly provided. (OE0791)</td></tr>
<tr><td>1. Perform a search for Standard C Library header files in the Core Framework source files. Verify that the mandatory header files are present. Record the source file name(s).</td><td>Pass: All mandatory header files are provided for within the context of the Core Framework (OE0791)

Fail: One or more mandatory header files are not defined within the context of the Core Framework. (OE0791)</td><td></td><td>Table B-25 (POSIX Standard C Library Header Files labeled MAN):<table><tr><td>ctype.h</td><td>limits.h</td></tr><tr><td>errno.h</td><td>signal.h</td></tr></table></td><td></td></tr>
<tr><td>2. Perform a search for Standard C Library header files in the Core Framework source files. Verify that the partially required header files are present. Record the source file name(s).</td><td>Pass: All partially required header files are provided for within the context of the Core Framework (OE0791)

Fail: One or more of the partially required header files are not defined within the context of the Core Framework. (OE0791)</td><td></td><td>Table B-25 (POSIX Standard C Library Header Files labeled PRT):<table><tr><td>locale.h</td><td>stdlib.h</td></tr><tr><td>math.h</td><td>string.h</td></tr><tr><td>stdarg.h</td><td>time.h</td></tr><tr><td>stdio’h</td><td></td></tr></table></td><td></td></tr>
<tr><td colspan="5">B. Verify that the mandatory functions listed in each Standard C Library header files, from Step A, are properly provided. (OE0791)</td></tr>
</table>

<table>
<tr><th colspan="5">OE_TC_175</th></tr>
<tr><th>Steps</th><th>Expected Results</th><th>Actual Results</th><th>Comments</th><th>Test Result</th></tr>
<tr><td>3. Verify that the mandatory functions listed in Table B-28 (ctype.h) are available in the Core Framework.</td><td>Pass: All mandatory header files are provided for within the context of the Core Framework (OE0791)

Fail: One or more mandatory header files are not defined within the context of the Core Framework. (OE0791)</td><td></td><td>.
Table B-28 (<ctype.h>):

<table><tr><td>int isalnum(int c)</td><td>int isalpha(int c)</td></tr><tr><td>int isblank(int c)</td><td>int iscntrl(int c)</td></tr><tr><td>int isdigit(int c)</td><td>int isgraph(int c)</td></tr><tr><td>int islower(int c)</td><td>int isprint(int c)</td></tr><tr><td>int ispunct(int c)</td><td>int isspace(int c)</td></tr><tr><td>int isupper(int c)</td><td>int isxdigit(int c)</td></tr><tr><td>int tolower(int c)</td><td>int toupper(int c)</td></tr></table></td><td></td></tr>
<tr><td>4. Verify that the mandatory functions listed in errno.h are available in the Core Framework.</td><td>N/A: There are no mandatory functions. (OE0791)</td><td></td><td><errno.h> contains 3 macros, namely, EDOM, EILSEQ, and ERANGE; but no mandatory functions.</td><td></td></tr>
<tr><td>5. Verify that the mandatory functions listed in limits.h are available in the OE header files.</td><td>N/A: There are no mandatory functions. (OE0791)</td><td></td><td><limits.h> contains several macros but no mandatory functions.</td><td></td></tr>
<tr><td>6. Verify that the mandatory functions listed in Table B-34 (signal.h) are available in the Core Framework.</td><td>Pass: All mandatory header files are provided for within the context of the Core Framework (OE0791)

Fail: One or more mandatory header files are not defined within the context of the Core Framework. (OE0791)</td><td></td><td>Table B-34 (<signal.h>):

void (*signal(int sig, void (*func)(int)))(int);
int raise(int sig);</td><td></td></tr>
<tr><td>7. Verify that the mandatory functions listed in Table B-31 (locale.h) are available in the OE header files.</td><td>Pass: All mandatory functions are provided for within the context of the Core Framework (OE0791)

Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)</td><td></td><td>Table B-31 (<locale.h>)

char *setlocale(int category,
const char *locale);</td><td></td></tr>
</table>

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

8. Verify that the mandatory functions listed in Table B-32 **(math.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)
	
	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)
			Table B-31 (**<math.h>)**
			
			double acos(double x);
			
			double asin(double x);
			
			double atan(double x);
			
			double atan2(double y, double x);
			
			double cos(double x);
			
			double sin(double x);
			
			double tan(double x);
			
			double cosh(double x);
			
			double sinh(double x);
			
			double tanh(double x);
			
			double exp(double x);
			
			double frexp(double value, int *exp);
			
			double ldexp(double x, int exp);
			
			double log(double x);
			
			double log10(double x);
			
			double modf(double value, double *iptr);
			
			double fabs(double x);	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

			double pow(double x, double y);
			double sqrt(double x);
			double ceil(double x);
			double floor(double x);
			double fmod(double x, double y);	

9. Verify that the mandatory functions listed in Table B-35 (**stdarg.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)
	
	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)
			Table B-35 (**<stdarg.h>)**
			void va_arg(va_list ap, *type*);
			void va_end(va_list ap);
			void va_start(va_list ap, *parmN*)**;**	

10. Verify that the mandatory functions listed in Table B-37 (**stdio.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)
	
	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)
			Table B-37 (**<stdio.h>)**
			int remove(const char * filename);
			int rename(const char *old, const char * new);
			int fclose(FILE * stream);
			int fflush(FILE * stream);
			FILE * fopen(const char * restrict filename, const char * restrict mode);
			FILE * freopen(const char * restrict filename,	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

			const char * restrict mode,FILE * restrict stream);
			
			void setbuf(FILE * restrict stream,
			char * restrict buf);
			
			int setvbuf(FILE * restrict stream,
			char * restrict buf, int mode, size_t size);
			
			int fprintf(FILE * restrict stream,
			const char * restrict format, ...);
			
			int fscanf(FILE * restrict stream,
			const char * restrict format, ...);
			
			int printf(const char * restrict format, ...);
			
			int snprintf(char * restrict s, size_t n,
			const char * restrict format, ...);
			
			int sscanf(const char * restrict s,
			const char * restrict format, ...);
			
			int fgetc(FILE *stream);
			
			char *fgets(char * restrict s, int n,
			FILE * restrict stream);
			
			int fputc(int c, FILE *stream);
			
			int fputs(const char * restrict s,
			FILE * restrict stream);
			
			int getc(FILE *stream);
			
			int getchar(void);	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

			int putc(int c, FILE *stream);
			
			int putchar(int c);
			
			int ungetc(int c, FILE *stream);
			
			size_t fread(void * restrict ptr,size_t size,
			size_t nmemb, FILE * restrict stream);
			
			size_t fwrite(const void * restrict ptr, size_t size,
			size_t nmemb, FILE * restrict stream);
			
			int fseek(FILE *stream, long int offset,
			int whence);
			
			long int ftell(FILE *stream);
			
			void rewind(FILE *stream);
			
			void clearerr(FILE *stream);
			
			int feof(FILE *stream);
			
			int ferror(FILE *stream);
			
			void perror(const char *s);
			
			int vsnprintf(char * restrict s, size_t n,
			const char * restrict format, va_list arg);	

11. Verify that the mandatory functions listed in Table B-38 (**stdlib.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)		Table B-38 (**<stdlib.h>)** double atof(const char *nptr);	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)		int atoi(const char *nptr);
			
			long int atol(const char *nptr);
			
			double strtod(const char * restrict nptr,
			char ** restrict endptr);
			
			long int strtol(const char* restrict nptr,
			char ** restrict endptr, int base);
			
			unsigned long int strtoul(
			const char * restrict nptr,
			char ** restrict endptr, int base);
			
			int rand(void);
			
			void srand(unsigned int seed);
			
			void *calloc(size_t nmemb, size_t size);
			
			void free(void *ptr);
			
			void *malloc(size_t size);
			
			void *realloc(void *ptr, size_t size);
			
			void abort(void);
			
			void *bsearch(const void *key,
			const void *base,size_t nmemb,
			size_t size,
			int (*compar)(const void *, const void *));
			
			void qsort(void *base, size_t nmemb,
			size_t size,	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

			int (*compar)(const void *, const void *));
			
			int abs(int j);
			
			long int labs(long int j);	

12. Verify that the mandatory functions listed in Table B-39 (**string.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)
	
	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)
			Table B-39 (**<string.h>)**
			
			void *memcpy(void * restrict s1,
			const void * restrict s2, size_t n);
			
			char *strncpy(char * restrict s1,
			const char * restrict s2, size_t n);
			
			char *strncat(char * restrict s1,
			const char * restrict s2, size_t n);
			
			int memcmp(const void *s1,
			const void *s2, size_t n);
			
			int strcmp(const char *s1, const char *s2);
			
			int strcoll(const char *s1, const char *s2);
			
			int strncmp(const char *s1,
			const char *s2, size_t n);
			
			size_t strxfrm(char * restrict s1,
			const char * restrict s2, size_t n);
			
			void *memchr(const void *s, int c, size_t n);
			
			char *strchr(const char *s, int c);
			
			size_t strcspn(const char *s1, const char *s2);	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

			char *strpbrk(const char *s1, const char *s2);
			
			char *strrchr(const char *s, int c);
			
			size_t strspn(const char *s1, const char *s2);
			
			char *strstr(const char *s1, const char *s2);
			
			char *strtok(char * restrict s1,
			const char * restrict s2);
			
			void *memset(void *s, int c, size_t n);
			
			size_t strlen(const char *s);
			
			void *memmove(void *s1, const void *s2,
			size_t n);
			
			char *strerror(int errnum);	

OE_TC_175				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**

13. Verify that the mandatory functions listed in Table B-41 (**time.h)** are available in the OE header files.	**Pass:** All mandatory functions are provided for within the context of the Core Framework (OE0791)
	
	Fail: One or more mandatory functions are not defined within the context of the Core Framework. (OE0791)
			Table B-41 (**<time.h>)**
			
			clock_t clock(void);
			
			time_t mktime(struct tm *timeptr);
			
			time_t time(time_t *timer);
			
			char *asctime(const struct tm *timeptr);
			
			char *ctime(const time_t *timer);
			
			struct tm *gmtime(const time_t *timer);
			
			struct tm *localtime(const time_t *timer);
			
			size_t strftime(char * restrict s,
			size_t maxsize,
			const char * restrict format,
			const struct tm * restrict timeptr);	

End of Test				

Test Recording Log – OE_TC_175		
Step 1 (MAN source file names)	**Step 3** (ctype functions exist-Y/N?)	**Step 6** (signal functions exist-Y/N?)

Test Recording Log – OE_TC_175							
Step1 (MAN source file names)						**Step3** (ctype functions exist-Y/N?)	**Step6** (signal functions exist-Y/N?)
Step2 (PTR source file names)	**Step7** (locale functions exist-Y/N?)	**Step8** (math functions exist-Y/N?)	**Step9** (stdarg functions exist-Y/N?)	**Step10** (stdio functions exist-Y/N?)	**Step11** (stdlib functions exist-Y/N?)	**Step12** (string functions exist-Y/N?)	**Step13** (time functions exist-Y/N?)

Test Summary OE_TC_175

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

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

OE0791 (ctype.h) _____

OE0791 (signal.h) _____

OE0791 (locale.h) _____

OE0791 (math.h) _____

OE0791 (stdarg.h) _____

OE0791 (stdio.h) _____

OE0791 (stdlib.h) _____

OE0791 (string.h) _____

OE0791 (time.h) _____

Failed Items (Section/Step Number):

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____