Joint Tactical Networking Center Test and Evaluation Laboratory

Software Communications Architecture v2.2.2
Manual Application Software Test Description v2.3A

Appendix C: AEP 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

Table of Contents

APPENDIX C AEP AMENDED REQUIREMENTS TEST CASES

This Appendix contains the test case procedures for the AEP Amended requirements. The test cases 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 Manual Test Steps table contains the test description as the section 'headers' followed by the more detailed manual test steps. The test description provides 'what' is being tested – not the 'how'. Each table row contains the steps of what the tester needs to do (such as examine, verify, locate, etc.). 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. When the tester needs additional space to record test results, they can use the Test Recording Log, explained below.
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 used to record test information. In addition, to being used as a test artifact, the Test Recording Log is useful for providing verification of test results. The Test Recording Log is also 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 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 _____

Application _____

C.1 APP_TC_047 - AEP Amended Ada functionality

Test Case Number: APP_TC_047

Application::Ada Functionality

Requirements

SCAv2.2.2 Tag	SCA v2.2.2A Text
AP0792	Any Ada application shall be restricted to using the equivalent Ada functionality, as defined in POSIX Ada language binding (ISO/IEC 14519:2001), designated as mandatory by the AEP or may use the C interface.

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B - Application Environment Profile (AEP)	AMENDED / 22 October 2008 Version 2.2.2A ,ICWG Approved>	Page 2, Section B.4
IEEE Standard for Information Technology - POSIX Ada Language Interfaces - Part 1: Binding for System Application Program Interface (API)	15 December 2001	Annex C Page 664-665 Section Package POSIX_Options (2.5)

Test Objective

This test case verifies AP0792. The objective of this test is to verify that if the application is using the Ada language then it is restricted to using equivalent Ada functionality as defined in the ISO/IEC 14519:2001 POSIX Ada Language Interface specification. Annex C of the IEEE document shows the Ada to C language cross references.

Places to Verify

Header files such as unistd.h

IDL References

None

Preconditions

- The application source code files having implementations of Ada and associated header files are available.

Test Description

For each application interface under test:
A. Verify that the interfaces implemented in Ada are restricted to using the equivalent Ada functionalities as referenced in the POSIX Ada Language Binding specification designated as mandatory in the SCA v2.2.2A Appendix B. (AP0792).

1. **Pass**: Equivalent Ada functionalities designated as mandatory in the SCA v2.2.2A Appendix B are implemented.
2. **Fail**: One or more equivalent Ada functionalities designated as mandatory in the SCA v2.2.2A Appendix B are not implemented.
3. **Fail:** Using an equivalent Ada function listed in Table B-2 as not mandatory.
4. **N/A:** No Ada functionality implemented.

List of mandatory POSIX option requirements seen in SCA v2.2.2A Appendix B, Table B-2:

_POSIX_ASYNCHRONOUS_IO
_POSIX_MEMLOCK_RANGE
_POSIX_MEMLOCK
_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

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.

APP_TC_047				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Verify that the interfaces implemented in Ada are restricted to using the equivalent Ada functionalities as referenced in the POSIX Ada Language Binding specification designated as mandatory in Appendix B. (AP0792).				
1. Examine the application interfaces having Ada implementations. List the source code file name(s).	**Pass**: Interfaces using equivalent Ada functionalities designated as mandatory in Appendix B. (AP0792) **Fail**: One or more interfaces not using the equivalent Ada functionalities designated as mandatory in Appendix B. (AP0792) **N/A:** No Ada functionality implemented.		A key word search using a semi-automated tool such as an IDE would be the easiest way to find the implementations of Ada.	
2. Verify that the Ada functions listed in Table B-2 as mandatory exist.	**Pass**: All the equivalent Ada functions listed in Table B-2 as mandatory exist. (AP0792) **Fail**: One or more equivalent Ada functions listed in Table B-2 as mandatory do not exist. (AP0792) **Fail**: Using an equivalent Ada function listed in Table B-2 as not mandatory. (AP0792)		**Table B-2 in Appendix B list the mandatory POSIX Ada language (Option requirements) with label** MAN.	
End of test				

Test Recording Log – APP_TC_047	
Step 1 (source file name)	**Step 2** (mandatory function exists-Y/N?)

Test Summary APP_TC_047

Once testing is complete for every component of the application 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):

AP0792 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

C.2 APP_TC_048 - AEP Amended Applications have no abnormal termination

Test Case Number: APP_TC_048

AEP abnormal termination

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
AP0798	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, SCA Application Environment Profile (AEP)	AMENDED / 22 October 2008 Version 2.2.A <ICWG Approved>	Page 5, Section B.4.1.4
SCA	Version 2.2.2 05-15-2006	Page 3-95, Section 3.2.1.1, paragraph 1
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 AP0798. Table B-6 of the SCA Appendix B, AEP Amended specifies the mandatory POSIX Signals function to be provided by the OE. This test case should **only** be executed if the waveform application provides any of the signals functions in Table B-6 in which the functions over-ride the base POSIX functions provided by the OE. The SCA AEP Amended specification requires that these signals functions when called must be handled properly so that other processes are not impacted. The objective of this test is to verify that the waveform application, which is the provider of these signals functions implements signal handlers for each signal function in order to properly manage process terminations in an environment that does not support multiple processes.

This test case replaces APP_TC_022 when testing is for SCA AEP Amended requirements.

Places to Verify

Application

IDL References

None

Preconditions

- The waveform application is the provider of one or more signals functions listed in Table B-6 of the SCA Appendix B, AEP Amended.
- Complete set of header files for the application environment are available.
- The application/waveform is to be verified against the AEP Amended requirements.

Test Description

A. Identify the source code and/or header files that implement POSIX Signal functions listed in Table B-6. (AP0798)
   1. **Pass:** The source code and/or header files are available.
   2. **Untested**: The source code files/ or header files are not available. If untested, then stop test.

For each POSIX Signal function identified in Step A, perform the following:

B. Verify there is a signal handler for each of the POSIX signal functions. (AP0798)
   1. **Pass:** All signal functions has a corresponding signal handler.
   2. **Fail:** One or more signal function(s) do not have a corresponding signal handler.

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.

APP_TC_048				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code and/or header files that implement POSIX Signal functions listed in Table B-6. (AP0798)				
1. Perform a search for header files for each signal function. Refer to list of functions in the Comments cell.	**Pass:** The source code and/or header files are available for all functions. (AP0798) **Untested**: The source code files are not available for one or more function(s). (AP0798)		abort() kill() pause() raise() sigaction() sigaddset() sigdelset() sigemptyset() sigfillset() sigismember() signal() sigpending() sigprocmask() sigsuspend() sigwait()	
For each POSIX Signal function identified in Step A, perform the following:				
B. Verify there is a signal handler for each of the POSIX signal functions. (AP0798)				

APP_TC_048				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
2. For each signal raised, verify that the corresponding signal is processed within the corresponding signal function.	**Pass:** Signal handlers are implemented for each POSIX signal. (AP0798) **Fail:** One or more signal(s) do not have a signal handler. (AP0798)		The OE is expected to safeguard the waveform application from any abnormal termination of processes when a signal function is called on a single process. Signal handlers are generally implemented to intercept the behavior of the signal (which is to interrupt a process' normal flow). If a process registers to a signal handler, then its routine is executed instead.	
End of test				

Test Recording Log – APP_TC_048	
Step 1 (source file having POSIX signal functions)	**Step 2** (List signal hander for each signal function.

Test Summary APP_TC_048

Once testing is complete for every component of the application 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):

AP0798 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

C.3 APP_TC_049 - AEP Amended kill function limitations

Test Case Number: APP_TC_049

Application Environment Profile::kill()

Requirements

SCA v2.2.2 Tag	SCA v2.2.2A Text
AP0799	An application that conforms to the AEP shall not call the kill() function with a negative argument because this profile does not require process group functionality.

References

Document Name	Version/Date	Location (Pages, Section)
SCA Appendix B, SCA Application Environment Profile (AEP)	AMENDED / 22 October 2008 Version 2.2.2A <ICWG Approved>	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 AP0799. The objective of this test is to verify that that the kill() function is not called with a negative argument. The kill() function uses two arguments, namely, pid and signal, to send a signal to a process or a group of processes specified by the pid (process ID). The second argument, signal, cannot be negative.

Places to Verify

Applications, Devices, and Services.

IDL References

None

Preconditions

- The application source code files are available.

Test Description

A. Identify the source code files that implement the kill() function. (AP0799)
   1. **Untested**: The source code files are not available.
B. Verify that the second argument when calling the kill() function is not negative. (AP0799)
   1. **Pass**: The second argument of the kill() function is not negative.
   2. **Fail**: The second argument of the kill() function is negative.

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.

APP_TC_049				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement the kill() function. (AP0799)				
1. Perform a search for the *kill* function in the source code provided by the developer.	**Untested**: The source code files are not available. (AP0799)		The kill() function will most likely be called on devices and services.	
B. Verify that the second argument when calling the kill() function is not negative. (AP0799)				
2. Verify that the 2nd argument when calling the kill function is not negative.	**Pass**: The second argument when calling the kill() function is not negative. (AP0799) **Fail**: The second argument when calling the kill() function is negative. (AP0799)		int CF_POSIX::kill(const int pid, **const int posixSignal**)	
End of test				

Test Recording Log – APP_TC_049	
Step 2 (component name having the kill function)	**Step 3** (Is 2nd argument a negative value– Y/N?)

Test Summary APP_TC_049

Once testing is complete for every component of the application 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):

AP0799 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____

C.4 APP_TC_050 - AEP Amended mandatory functions

Test Case Number: APP_TC_050

AEP Amended OS Services

Requirements

SCA v2.2.2 Tag	SCA v2.2.2 Text
AP0603	Applications shall be limited to using the OS services that are designated as mandatory in the SCA Application Environment Profile (Appendix B).

References

Document Name	Version/Date	Location (Pages, Section)
SCA	Final 15 May 2006 Version 2.2.2	Page 3-95, Section 3.2.1.1
SCA Appendix B, SCA Application Environment Profile (AEP)	AMENDED / 22 October 2008 Version 2.2.2A <ICWG Approved>	Page 11-12, Section B.4.1.15
SCA Networking AEP	Version 2.2.2a, 19 March 2010	Pages 1 to 8
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 6, Section 1.5, Table 1-1

Test Objective

This test case verifies AP0603. The objective of the test is to verify that the application is limited to using services that are designated mandatory in the SCA v2.2.2 Application Environment Profile (AEP), version 2.2.2A and the SCA Networking AEP.

Places To Verify

Application source code files.

IDL References

None.

Preconditions

- The application source code files are available.
- The Application requires the verification of the SCA AEP Amended requirements.

Test Description

Note: The best way to verify AEP compliance is to prove that violations to the requirement AP0603 do not exist. Below is a table of AEP OS service violations as compiled by JTEL.

For each component, perform the following:

A. Identify the source code files that implement POSIX OS service functions. (AP0603)
   1. **N/A**: The source code files are not available.

For each entry in the APP_TC_050 Table 1:

B. Search the source code for occurrences of that entry. (AP0603)
   1. **Pass**: There were no occurrences of the table entry. Move on to the next table entry.

C. Research occurrences within the source code to verify that they are not implementations of a POSIX OS service interface. (AP0603)
   1. **Pass**: All occurrences represent other than implementations of POSIX interfaces.
   2. **Fail**: Some occurrence represents the implementation of a POSIX interface.

The following is a table of AEP POSIX OS service calls that would violate requirement AP0603. They are to be used to perform steps B and C above.

APP_TC_050 Table 1			
List of violations of AEP Requirements			
_Exit	flockfile	mkstemp	sched_setscheduler
_exit	floorf	mktemp	sched_yield
_longjmp	floorl	mmap	seed48
_setjmp	fma	modff	seekdir
_tolower	fmaf	modfl	sem_timedwait
_toupper	fmal	mprotect	semctl
a64l	fmax	mq_timedreceive	semget
acosf	fmaxf	mq_timedsend	semop
acosh	fmaxl	mrand48	sendmsg
acoshf	fmin	msgctl	setcontext

APP_TC_050 Table 1			
List of violations of AEP Requirements			
acoshl	fminf	msgget	setegid
acosl	fminl	msgrcv	setenv
alarm	fmodf	msgsnd	seteuid
asinf	fmodl	msync	setgid
asinh	fmtmsg	munmap	setgrent
asinhf	fnmatch	nan	sethostent
asinhl	fork	nanf	setitimer
asinl	fpclassify	nanl	setjmp
assert	fputwc	nearbyint	setkey
atan2f	fputws	nearbyintf	setlogmask
atan2l	freeaddrinfo	nearbyintl	setnetent
atanf	frexpf	nextafter	setpgid
atanh	frexpl	nextafterf	setpgrp
atanhf	fsetpos	nextafterl	setpriority
atanhl	fstatvfs	nexttoward	setprotoent
atanl	fsync	nexttowardf	setpwent
atexit	ftime	nexttowardl	setregid
atoll	ftok	nftw	setreuid
basename	ftruncate	nice	setrlimit
bcmp	ftrylockfile	nl_langinfo	setservent
bcopy	ftw	nrand48	setsid
bsd_signal	funlockfile	openlog	setstate
btowc	fwide	optarg	setuid
bzero	fwprintf	opterr	setutxent
cabs	fwscanf	optind	shm_open
cabsf	gai_strerror	optopt	shm_unlink
cabsl	gcvt	pclose	shmat
cacos	getaddrinfo	pipe	shmctl
cacosf	getc_unlocked	poll	shmdt
cacosh	getchar_unlocked	popen	shmget
cacoshf	getcontext	posix_fadvise	shutdown
cacoshl	getdate	posix_fallocate	sigaltstack

APP_TC_050 Table 1			
List of violations of AEP Requirements			
cacosl	getdate_err	posix_madvise	sighold
carg	getegid	posix_mem_offset	sigignore
cargf	getenv	posix_memalign	siginterrupt
cargl	geteuid	posix_openpt	siglongjmp
casin	getgid	posix_spawn	signbit
casinf	getgrent	posix_spawn_file_actions_addclose	signgam
casinh	getgrgid	posix_spawn_file_actions_adddup2	sigpause
casinhf	getgrgid_r	posix_spawn_file_actions_addopen	sigrelse
casinhl	getgrnam	posix_spawn_file_actions_destroy	sigset
casinl	getgrnam_r	posix_spawn_file_actions_init	sigsetjmp
catan	getgroups	posix_spawnattr_destroy	sinf
catanf	gethostbyaddr	posix_spawnattr_getflags	sinhf
catanh	gethostbyname	posix_spawnattr_getpgroup	sinhl
catanhf	gethostent	posix_spawnattr_getschedparam	sinl
catanhl	gethostid	posix_spawnattr_getschedpolicy	sleep
catanl	gethostname	posix_spawnattr_getsigdefault	sockatmark
catclose	getitimer	posix_spawnattr_getsigmask	socketpair
catgets	getlogin	posix_spawnattr_init	sprintf
catopen	getlogin_r	posix_spawnattr_setflags	sqrtf
cbrt	getmsg	posix_spawnattr_setpgroup	sqrtl
cbrtf	getnameinfo	posix_spawnattr_setschedparam	srand48
cbrtl	getnetbyaddr	posix_spawnattr_setschedpolicy	srandom
ccos	getnetbyname	posix_spawnattr_setsigdefault	statvfs
ccosf	getnetent	posix_spawnattr_setsigmask	strcasecmp
ccosh	getopt	posix_spawnp	strcat
ccoshf	getpeername	posix_trace_attr_destroy	strcpy
ccoshl	getpgid	posix_trace_attr_getclockres	strdup
ccosl	getpgrp	posix_trace_attr_getcreatetime	strfmon
ceilf	getpid	posix_trace_attr_getgenversion	strncasecmp
ceill	getpmsg	posix_trace_attr_getinherited	strptime
cexp	getppid	posix_trace_attr_getlogfullpolicy	strtof
cexpf	getpriority	posix_trace_attr_getlogsize	strtoimax

APP_TC_050 Table 1			
List of violations of AEP Requirements			
cexpl	getprotobyname	posix_trace_attr_getmaxdatasize	strtold
cfgetispeed	getprotobynumber	posix_trace_attr_getmaxsystemeventsize	strtoll
cfgetospeed	getprotoent	posix_trace_attr_getmaxuserеventsize	strtoull
cfsetispeed	getpwent	posix_trace_attr_getname	strtoumax
cfsetospeed	getpwnam	posix_trace_attr_getstreamfullpolicy	swab
chmod	getpwnam_r	posix_trace_attr_getstreamsize	swapcontext
chown	getpwuid	posix_trace_attr_init	swprintf
cimag	getpwuid_r	posix_trace_attr_setinherited	swscanf
cimagf	getrlimit	posix_trace_attr_setlogfullpolicy	symlink
cimagl	getrusage	posix_trace_attr_setlogsize	sync
clock	gets	posix_trace_attr_setmaxdatasize	SynchronousReceive
clock_getcpuclockid	getservbyname	posix_trace_attr_setname	SynchronousSend
clock_nanosleep	getservbyport	posix_trace_attr_setstreamfullpolicy	sysconf
clog	getservent	posix_trace_attr_setstreamsize	syslog
clogf	getsid	posix_trace_clear	system
clogl	getsockname	posix_trace_close	tanf
closelog	getsubopt	posix_trace_create	tanhf
confstr	gettimeofday	posix_trace_create_withlog	tanhl
conj	getuid	posix_trace_event	tanl
conjf	getutxent	posix_trace_eventid_equal	tcdrain
conjl	getutxid	posix_trace_eventid_get_name	tcflow
copysign	getutxline	posix_trace_eventid_open	tcflush
copysignf	getwc	posix_trace_eventset_add	tcgetattr
copysignl	getwchar	posix_trace_eventset_del	tcgetpgrp
cosf	getwd	posix_trace_eventset_empty	tcgetsid
coshf	glob	posix_trace_eventset_fill	tcsendbreak
coshl	globfree	posix_trace_eventset_ismember	tcsetattr
cosl	grantpt	posix_trace_eventtypelist_getnext_id	tcsetpgrp
cpow	HaltTask	posix_trace_eventtypelist_rewind	tdelete
cpowf	hcreate	posix_trace_flush	telldir
cpowl	hdestroy	posix_trace_get_attr	tempnam
cproj	hsearch	posix_trace_get_filter	tfind

APP_TC_050 Table 1			
List of violations of AEP Requirements			
cprojf	hypot	posix_trace_get_status	tgamma
cprojl	hypotf	posix_trace_getnext_event	tgammaf
creal	hypotl	posix_trace_open	tgammal
crealf	iconv	posix_trace_rewind	TimedSynchronousReceive
creall	iconv_close	posix_trace_set_filter	TimedSynchronousSend
crypt	iconv_open	posix_trace_shutdown	times
csin	if_freenameindex	posix_trace_start	timezone
csinf	if_indextoname	posix_trace_stop	tmpfile
csinh	if_nameindex	posix_trace_timedgetnext_event	tmpnam
csinhf	if_nametoindex	posix_trace_trid_eventid_open	toascii
csinhl	ilogb	posix_trace_trygetnext_event	towctrans
csinl	ilogbf	posix_typed_mem_get_info	towlower
csqrt	ilogbl	posix_typed_mem_open	towupper
csqrtf	imaxabs	powf	trunc
csqrtl	imaxdiv	powl	truncate
ctan	index	pread	truncf
ctanf	inet_addr	pselect	truncl
ctanh	inet_ntoa	pthread_atfork	tsearch
ctanhf	inet_ntop	pthread_barrier_destroy	ttyname
ctanhl	inet_pton	pthread_barrier_init	ttyname_r
ctanl	initstate	pthread_barrier_wait	twalk
ctermid	insque	pthread_barrierattr_destroy	tzname
CurrentTask	ioctl	pthread_barrierattr_getpshared	tzset
daylight	isascii	pthread_barrierattr_init	ualarm
dbm_clearerr	isastream	pthread_barrierattr_setpshared	ulimit
dbm_close	isatty	pthread_condattr_getclock	umask
dbm_delete	isfinite	pthread_condattr_getpshared	uname
dbm_error	isgreater	pthread_condattr_setclock	ungetwc
dbm_fetch	isgreaterequal	pthread_condattr_setpshared	unlockpt
dbm_firstkey	isinf	pthread_getconcurrency	unsetenv
dbm_nextkey	isless	pthread_getcpuclockid	usleep
dbm_open	islessequal	pthread_mutex_timedlock	utimes

APP_TC_050 Table 1			
List of violations of AEP Requirements			
dbm_store	islessgreater	pthread_mutexattr_getpshared	va_copy
difftime	isnan	pthread_mutexattr_setpshared	vfork
dirname	isnormal	pthread_rwlock_destroy	vfprintf
div	isunordered	pthread_rwlock_init	vfscanf
dlclose	iswalnum	pthread_rwlock_rdlock	vfwprintf
dlerror	iswalpha	pthread_rwlock_timedrdlock	vfwscanf
dlopen	iswblank	pthread_rwlock_timedwrlock	vprintf
dlsym	iswcntrl	pthread_rwlock_tryrdlock	vscanf
drand48	iswctype	pthread_rwlock_trywrlock	vsprintf
dup	iswdigit	pthread_rwlock_unlock	vsscanf
dup2	iswgraph	pthread_rwlock_wrlock	vswprintf
ecvt	iswlower	pthread_rwlockattr_destroy	vswscanf
encrypt	iswprint	pthread_rwlockattr_getpshared	vwprintf
endgrent	iswpunct	pthread_rwlockattr_init	vwscanf
endhostent	iswspace	pthread_rwlockattr_setpshared	wait
endnetent	iswupper	pthread_setconcurrency	waitid
endprotoent	iswxdigit	pthread_setschedprio	waitpid
endpwent	j0	pthread_spin_destroy	wcrtomb
endservent	j1	pthread_spin_init	wcscat
endutxent	jn	pthread_spin_lock	wcschr
environ	jrand48	pthread_spin_trylock	wcscmp
erand48	killpg	pthread_spin_unlock	wcscoll
erf	l64a	ptsname	wcscpy
erfc	lchown	putc_unlocked	wcscspn
erfcf	lcong48	putchar_unlocked	wcsftime
erfcl	ldexpf	putenv	wcslen
erff	ldexpl	putmsg	wcsncat
erfl	ldiv	putpmsg	wcsncmp
execl	lfind	puts	wcsncpy
execle	lgamma	pututxline	wcspbrk
execlp	lgammaf	putwc	wcsrchr
execv	lgammal	putwchar	wcsrtombs

APP_TC_050 Table 1			
List of violations of AEP Requirements			
execve	llabs	pwrite	wcsspn
execvp	lldiv	random	wcsstr
exit	llrint	readlink	wcstod
Exit	llrintf	readv	wcstof
exp2	llrintl	realpath	wcstoimax
exp2f	llround	recvmsg	wcstok
exp2l	llroundf	regcomp	wcstol
expf	llroundl	regerror	wcstold
expl	localeconv	regexec	wcstoll
expm1	lockf	regfree	wcstombs
expm1f	log10f	ReleaseResource	wcstoul
expm1l	log10l	remainder	wcstoull
fabsf	log1p	remainderf	wcstoumax
fabsl	log1pf	remainderl	wcswcs
fattach	log1pl	remque	wcswidth
fchdir	log2	remquo	wcsxfrm
fchmod	log2f	remquof	wctob
fchown	log2l	remquol	wctomb
fcntl	logb	RequestResource	wctrans
fcvt	logbf	rindex	wctype
FD_CLR	logbl	rint	wcwidth
FD_ISSET	logf	rintf	wmemchr
FD_SET	logl	rintl	wmemcmp
FD_ZERO	longjmp	round	wmemcpy
fdatasync	lrand48	roundf	wmemmove
fdetach	lrint	roundl	wmemset
fdim	lrintf	scalb	wordexp
fdimf	lrintl	scalbln	wordfree
fdiml	lround	scalblnf	wprintf
feclearexcept	lroundf	scalblnl	writev
fegetenv	lroundl	scalbn	wscanf
fegetexceptflag	lsearch	scalbnf	y0

APP_TC_050 Table 1			
List of violations of AEP Requirements			
fegetround	lstat	scalbnl	y1
feholdexcept	makecontext	scanf	yn
feraiseexcept	mblen	sched_get_priority_max	
fesetenv	mbrlen	sched_get_priority_min	
fesetexceptflag	mbrtowc	sched_getparam	
fesetround	mbsinit	sched_getscheduler	
fetestexcept	mbsrtowcs	sched_rr_get_interval	
feupdateenv	mbstowcs	sched_setparam	
ffs	mbtowc		
fgetpos	memccpy		
fgetwc	mkfifo		
fgetws	mknod		

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.

APP_TC_050				
Steps	**Expected Results**	**Actual Results**	**Comments**	**Test Result**
A. Identify the source code files that implement an OS service function. (AP0603)				
1. Perform a search for all OS service functions in the application source code provided by the developer. Record the source file name(s).	**N/A**: The source code files are not available. (AP0603)			
For each entry in the APP_TC_050 Table 1 perform steps 2 & 3 :				
B. Search the source code for occurrences of that entry. (AP0603)				
2. Using the select entry from Table 1, perform a search for that entry.	**Pass:** There are no occurrences of the table entry. Move on to the next table entry. (AP0603) Otherwise go to Step 3			
C. Research occurrences within the source code to verify that they are not implementations of a POSIX interface. (AP0603)				
3. Research occurrences within the source code to verify that they are not implementations of a POSIX interface.	**Pass:** All occurrences of the table entry found in the search do not represent implementations of a POSIX interface. (AP0603) **Fail:** One or more occurrences of the table entry found in the search represent implementations of a POSIX interface. (AP0603)			
End of test				

Test Recording Log – APP_TC_050		
Step 1 (source file having OS service function calls)	**Step 2** (table entries that are present)	**Step 3** (Result of research of AEP OS violation)

Test Summary APP_TC_050

Once testing is complete for every component of the application under test, report for each requirement 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):

AP0603 _____

Failed Items (Section/Step Number):

_____

_____

_____

Test Engineer: _____

Date Tested: _____

Witness: _____