X/Open and XPG4 6

This chapter discusses the compliance of Solaris 2.5 to the programming interface specifications contained in the X/Open Portability Guide Issue 4, (XPG4).

The X/Open Portability Guide, Issue 4

In 1992, X/Open published the X/Open Portability Guide, Issue 4. XPG4 retains compliance to the POSIX.1–1988 standard but is extended to the ISO/IEC updated POSIX.1 standard and the ISO/IEC C language standard. The X/Open Portability Guide Issue 3 remains available and system and components can still be branded to XPG3, but XPG4 branding offers significant additional capability.

This chapter identifies Solaris 2.5 as a conforming implementation of XPG4, and displays the XPG4 Base brand trademark. It also presents the X/Open Conformance Statement, which documents Solaris' compliance to the programming interface specifications of the X/Open Portability Guide, Issue 4.

The X/Open Brand Trademark

X/Open provides a verification and branding program that developers can use to show that their products are X/Open compliant. Sun Microsystems has been a strong supporter of the X/Open branding process since its inception.

6

Figure 6-1 The XPG4 Base Brand Logo

Solaris is available as an XPG4 and/or XPG3 BASE branded configuration. Solaris products and software products from independent software vendors that have received XPG4 branding are described below:

- **Window Management (OpenWindows)**—The Solaris window system, which supports the OPEN LOOK Graphical User Interface, has earned the XPG4 brand by implementing the programmer's interface to the X Window System. OpenWindows supports the Window Management component of the X/Open BASE level.
- **Internationalized System Calls and Libraries**— Solaris meets the Internationalized System Calls and Libraries component requirements of the BASE level.
- **Commands and Utilities**—X/Open's specification of standard interfaces for utilities allows for portable shell scripts. Solaris meets the Commands and Utilities component requirements of the BASE system.
- **ProCompiler™ C 2.0.1**— The ProCompiler for Solaris for x86 is fully conformant with the ANSI/ISO Standard for C. It has passed X/Open verification test suite VSX 4.3.2 and meets the C Language component requirements of the BASE level.
- **SPARCompiler C 2.0.1**—The SPARCompiler for the C programming language based on Common Usage C has passed X/Open verification test suite VSX 4.3.2 and meets the C Language component requirements of the BASE level.
- **Sun FORTRAN 3.0**—Sun's compiler for the FORTRAN programming language is fully compliant with the definition in the American National Standards Institute (ANSI) document.
- **Sun Pascal 3.0.1**—Sun's compiler for the Pascal programming language is fully compliant with the ISO standard.

- **Magnetic Media (Source Code Transfer)**—Sun conforms to X/Open's specifications for transferring source code between machines with compatible media and facilitating the transfer of source code in machine-readable form. Solaris supports the Source Code Transfer component of the X/Open BASE level.
- **Inter-Process Communication**—Sun supports X/Open's specifications for interfaces providing message queue, semaphore, and shared memory facilities for communication and synchronization between processes. The SunOS operating system fulfills the requirements of the Inter-Process Communications component of the X/Open BASE level.
- **Terminal Interfaces (XSI Curses Interface)**—The XSI Curses Interface meets X/Open's specifications for providing a generic terminal interface that is independent of terminal hardware or connection methods for updating screens on character-oriented and block-oriented terminals. Solaris systems fulfill the requirements of the Terminal Interfaces component of the X/Open BASE level.

The X/Open Conformance Statement for Solaris

The remaining pages of this chapter feature the X/Open Conformance Statement for Solaris.

X/Open Conformance Statement

X/OPEN Conformance Statement Questionnaire

1. Internationalized System Calls and Libraries

Product Identification

Product Identification	Solaris
Version/Release No.	2.5

If you do not supply this component yourself, please identify below the supplier you reference.

6

Indicator of Compliance

VSX Test Suite Release	VSX 4.3.4
Testing Agency Name	Mindcraft, Inc.
Address	410 Cambridge Avenue Palo Alto, California 94306

Environment Specification

Enter below details of the hardware and software environment in which testing took place, including compilation routines and installation procedures (if any). Sufficient detail must be supplied to enable conformant behavior and any test results to be reproduced.

SPARC

SPARC running Solaris 2.5. Installation procedures are provided in SPARC: Installing Solaris Software

To reproduce the test environment, do these steps:

1. Edit the `/etc/saf/zsmon/_pmtab` file to turn off the ttysoftcarrier detect:

   Change the `ttya` and `ttyb` fields from `:y:` to `:n:`. (The colons (:) act as field separators.)

2. Verify that the ttymodes settings in the `/kernel/drv/options.conf` file are set to:

```
2502:1805:bd:8a3b:3:1c:7f:15:4:0:0:0:11:13:1a:19:12:f:17:16
```

3. Disable `ypbind` to allow rebooting of the system:

   a. `cd /usr/lib/netsvc/yp`

   b. `mv ypbind ypbind-`

4. Set the `eeprom` variables that affect the `tty`:

   a. On the keyboard, hit `STOP-A` to display the `prom` prompt.

   b. At the prompt, execute the following steps:

```
setenv ttya-ignore-cd false
setenv ttyb-ignore-cd false
setenv ttya-rts-dtr-off false
setenv ttyb-rts-dtr-off false
```

5. Reboot the system

Note – When installing Solaris, set the time zone by selecting a time zone format that conforms to the POSIX.1 format for TZ defined on page 152 and page 153 of the IEEE Std. 1003.1–1990.

Note – The following option must be added to any compiler line:
`-D_XOPEN_VERSION=4`

Note – The following must be added to any link/load line:
`/usr/ccs/lib/values_xpg4.o`

---

x86

x86 running Solaris 2.5. Installation procedures are provided in x86: Installing Solaris Software.

To reproduce the test environment, follow these steps:

1. Become `root`.

2. Ensure the correct serial port links:

- `/dev/ttya` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/term/a` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/tty00` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/ttyb` should be a link to `/devices/isa/asy@2f8,0:a`
- `/dev/term/b` should be a link to `/devices/isa/asy@2f8,0:a`
- `/dev/tty01` should be a link to `/devices/isa/asy@2f8,0:b`

   a. If the `/dev/tty01` link is missing, perform the following:

- Edit `/kernel/drv/asy.conf` and uncomment the COM2 entry
- `# touch /reconfigure`

3. Set the correct serial port permissions:

- `# chmod 666 /devices/eisa/asy*`

4. Turn off the `ttysoftcarrier` detect:
   Using an editor such as `vi`, in the `/etc/saf/zsmon/_pmtab` file, change the next to last field for both the `ttya` entry and the `ttyb` entry from `y` to `n` (the colon (`:`) acts as the field separator):

- `# vi /etc/saf/zsmon/_pmtab`

5. Reboot the system.

Note – When installing Solaris, set the time zone by selecting a time zone format that conforms to the POSIX.1 format for TZ defined on page 152 and page 153 of the IEEE Std. 1003.1–1990.

Note – The following option must be added to any compiler line:
-D_XOPEN_VERSION=4

Note – The following must be added to any link/load line:
/usr/ccs/lib/values_xpg4.o

Temporary Waivers

None.

Section 1.1: General Attributes

1.1.1 XPG4 Feature Groups

Question 1: *Which of the following Feature Groups are supported by the implementation?*

Response:

POSIX C–language Binding	Yes
Shared Memory	Yes
Encryption	Yes
Enhanced Internationalization	Yes

Note: All the interfaces in all these groups must exist on all XPG4 XSI-conformant systems, and each interface must either behave according to the description in System Interfaces and Headers, Issue 4, or indicate an error, with *errno* set to [ENOSYS].

Support for particular Feature Groups may be required in order to conform to particular X/Open Profiles. Support for a Feature Group can only be claimed if *all* interfaces in that group behave according to the relevant descriptions in System Interfaces and Headers, Issue 4.

Rationale

System Interfaces and Headers, Issue 4 states that the system may provide one or more of the Feature Groups listed.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Section 1.2, Conformance and Section 1.3, *Feature Groups*.

1.1.2 POSIX.1 Supported Features

Question 2 *Which of the following options, specified in the* `<unistd.h>` *header file, are available on the system?*

Response:

Macro Name	Meaning	Provided
_POSIX_CHOWN_RESTRICTED	The use of chown() is restricted to a process with appropriate privileges and to changing the group ID of a file only to the effective group ID of the process or one of its supplementary groups IDs.	Yes
_POSIX_NO_TRUNC	Pathname components longer than {NAME_MAX} generate an error	Yes

(continued)

Macro Name	Meaning	Provided
_POSIX_VDISABLE	Terminal special characters defined in <termios.h> can be disabled using this character value.	Yes
_POSIX_SAVED_IDS	Each process has a saved set-user-ID and saved set-group-ID.	Yes
_POSIX_JOB_CONTROL	Implementation supports job control.	Yes

Rationale

For an XSI–conformant implementation, all of these POSIX features must be provided. In some cases the feature need not be provided for all files or devices supported by the implementation.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, Headers <unistd.h>.

1.1.3 Float, Stdio and Limit Values

Question 3: *What are the values associated with the following constants specified in the* **<float.h>** *header file?*

Response:

Macro Name	Meaning	Value
FLT_RADIX	Radix of the exponent representation.	2
FLT_MANT_DIG	Number of base-FLT_RADIX digits in the float significand.	24
DBL_MANT_DIG	Number of base-FLT_RADIX digits in the double significand.	53
LDBL_MANT_DIG	Number of base_FLT_RADIX digits in the long double significand.	64

(continued)

Macro Name	Meaning	Value
FLT_DIG	Number of decimal digits, q, such that any floating point number with q digits can be rounded into a float representation and back again without change to the q digits.	6
DBL_DIG	Number of decimal digits, q, such that any floating point number with q digits can be rounded into a double representation and back again without change to the q digits.	15
LDBL_DIG	Number of decimal digits, q, such that any floating point number with q digits can be rounded into a long double representation and back again without change to the q digits.	18
FLT_MIN_EXP	Minimum negative integer such that FLT_RADIX raised to that power minus 1 is a normalized float.	-125
DBL_MIN_EXP	Minimum negative integer such that FLT_RADIX raised to that power minus 1 is a normalized double.	-1024
LDBL_MIN_EXP	Minimum negative integer such that FLT_RADIX raised to that power minus 1 is a normalized long double.	-16381
FLT_MIN_10_EXP	Minimum negative integer such that 10 raised to that power is in the range of normalized floats.	-125
DBL_MIN_10_EXP	Minimum negative integer such that 10 raised to that power is in the range of normalized doubles.	-307
LDBL_MIN_10_EXP	Minimum negative integer such that 10 raised to that power is in the range of normalized long doubles.	-4931

(continued)

Macro Name	Meaning	Value
FLT_MAX_EXP	Maximum integer such that FLT_RADIX raised to that power minus 1 is a representable finite float.	128
DBL_MAX_EXP	Maximum integer such that FLT_RADIX raised to that power minus 1 is a representable finite double.	1024
LDBL_MAX_EXP	Maximum integer such that FLT_RADIX raised to that power minus 1 is a representable finite long double.	16384
FLT_MAX_10_EXP	Maximum integer such that 10 raised to that power is in the range of representable finite floats.	38
DBL_MAX_10_EXP	Maximum integer such that 10 raised to that power is in the range of representable finite doubles.	308
LDBL_MAX_10_EXP	Maximum integer such that 10 raised to that power is in the range of representable finite long doubles.	4932
FLT_MAX	Maximum representable finite float.	3.402823466E+38F
DBL_MAX	Maximum representable finite double.	1.7976931348623157E+308
LDBL_MAX	Maximum representable finite long double.	1.189731495357231765085759326628007016E+4932L
FLT_EPSILON	Difference between 1.0 and the least value greater than 1.0 that is representable as a float.	1.192092896E-07F
DBL_EPSILON	Difference between 1.0 and the least value greater than 1.0 that is representable as a double.	2.2204460492503131E-16

(continued)

Macro Name	Meaning	Value
LDBL_EPSILON	Difference between 1.0 and the least value greater than 1.0 that is representable as a long double.	1.92592994438723585305597794258492731 9E-34L
FLT_MIN	Minimum normalized positive float.	1.175494351E-38F
DBL_MIN	Minimum normalized positive double.	2.2250738585072014E-308
LDBL_MIN	Minimum normalized positive long double.	3.36210314311209350626267781732175226 03E-4932L

Rationale

This set of constants provides useful information regarding the underlying architecture of the implementation.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, *Headers*, **<float.h>**.

Question 4 *What are the values associated with the following constants (optionally specified in the* <**limits.h**> *header file)?*

Response:

Macro Name	Meaning	Minimum	Maximum
ARG_MAX	Maximum length of argument to the `exec` functions including the environment data.	4096	1048320
CHILD_MAX	Maximum number of processes per user ID.	6	See note
LINK_MAX	Maximum number of links to a single file.	8	32767
MAX_CANON	Maximum number of bytes in a terminal canonical input line.	255	256

(continued)

Macro Name	Meaning	Minimum	Maximum
NAME_MAX	Maximum number of bytes allowed in a terminal input queue.	14	See note
OPEN_MAX	Maximum number of open files that one process can have open at any one time.	16	See note
PATH_MAX	Maximum number of bytes in a pathname (including the terminating null).	255	See note
PIPE_BUF	Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.	512	5120
STREAM_MAX	Number of streams one process can have open at one time.	8	8
TZ_NAME_MAX	Maximum number of bytes supported for the name of a time zone.	3	3

Notes:
CHILD_MAX depends on how the system kernel is configured.

The maximum values for NAME_MAX and PATH_MAX vary depending on the file system type, but always provide at least the minimum requirement. The most common values are 255 `for` NAME_MAX and 1024 for PATH_MAX. Values for a specific path are available using pathconf().

OPEN_MAX defaults to 64, but users can increase or decrease this value using routines not specified by POSIX.1 or XPG4.

Rationale

Each of these limits can vary within bounds set by the Systems Interfaces and Headers, Issue 4. The minimum value that a limit can take on any XSI conforming system is given in the corresponding _POSIX_ value. A specific conforming implementation may provide a higher minimum value than this and the maximum value that it provides can differ from the minimum. Some conforming implementations may provide a potentially infinite value as the

maximum, in which case the value is considered to be indeterminate. The minimum value must always be definitive since the _POSIX_ value provides a known lower bound for the range of possible values.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, *Headers*, <limits.h>

Question 5: *What are the values associated with the following constants specified in the* **<limits.h>** *header file?*

Response:

Macro Name	Meaning	Min.	Maximum
BC_BASE_MAX	Maximum *ibase* and *obase* values allowed by the `bc` utility.	99	32767 (`SHRT_MAX`)
BC_DIM_MAX	Maximum number of elements permitted in an array by the `bc` utility.	2048	32768 (`SHRT_MAX+1`)
BC_SCALE_MAX	Maximum scale value allowed by the `bc` utility.	99	32767 (`SHRT_MAX`)
BC_STRING_MAX	Maximum length of a string constant accepted by the `bc` utility.	1000	2048 (`LINE_MAX`)
COLL_WEIGHTS_MAX	Maximum number of weights that can be assigned to an entry of the LC_COLLATE order keyword in the local definition file.	2	4
EXPR_NEST_MAX	Maximum number of expressions that can be nested within parentheses by the `expr` utility.	32	32
LINE_MAX	Maximum length in bytes including the trailing newline of a utility's input line when the utility is described as processing text files.	2048	2048

(continued)

Macro Name	Meaning	Min.	Maximum
NGROUPS_MAX	Maximum number of simultaneous supplementary group IDs per process.	16	16
RE_DUP_MAX	Maximum number of repeated occurrences of a regular expression permitted when using interval notation.	255	255

Rationale

Each of these limits can vary within bounds set by the System Interfaces and Headers, Issue 4. The minimum value that a limit can take on any XSI conforming system is given in the corresponding _POSIX_ or POSIX2_ value. A specific conforming implementation may provide a higher minimum value than this and the maximum value that it provides can differ from the minimum. Some conforming implementations may provide a potentially infinite value as the maximum, in which case the value is considered to be indeterminate. The minimum value must always be definitive since the _POSIX_ or _POSIX2_ value provides a known lower bound for the range of possible values.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, *Headers*, `<limits.h>`

Question 6:*What are the values associated with the following numerical constants specified in the* `<limits.h>` *header file?*

Response:

Macro Name	Meaning	Value
CHAR_BIT	Number of bits in a char.	8
CHAR_MAX	Maximum value of a char.	127
INT_MAX	Maximum value of an int.	2147483647

(continued)

Macro Name	Meaning	Value
LONG_BIT	Number of bits in a long int.	32
LONG_MAX	Maximum value of a long int.	2147483647L
MB_LEN_MAX	Maximum number of bytes in a character, for any supported locale.	5
SCHAR_MAX	Maximum value of a signed char.	127
SHRT_MAX	Maximum value of a short.	32767
SSIZE_MAX	Maximum value of an object of type ssize_t.	2147483647
UCHAR_MAX	Maximum value of an unsigned char.	25
UINT_MAX	Maximum value of an unsigned int.	4294967295U
ULONG_MAX	Maximum value of an unsigned long int.	4294967295UL
USHRT_MAX	Maximum value of an unsigned short int.	65535
WORD_BIT	Number of bits in a word or int.	32

Rationale

This set of constants provides useful information regarding the underlying architecture of the implementation.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, *Headers*, **<limits.h>.**

Question 7: *What are the values associated with the following numerical constants specified in the* **<stdio.h>** *header file?*

Response:

Macro Name	Meaning	Value
FOPEN_MAX	Number of streams which the implementation guarantees can be open simultaneously.	20 (SPARC) 60 (x86)
L_tmpnam	Maximum size of character array to hold tmpnam() output.	25
TMP_MAX	Minimum number of unique filenames generated by tmpnam(), which is the maximum number of times an application can call tmpnam() reliably.	17567

Rationale

This set of constants provides useful information about the implementation.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 4, *Headers*, **<stdio.h>**

1.1.4 Error Conditions

Question 8: *Which of the following option errors listed in the System Interfaces and Headers, Issue 4 are detected in the circumstances specified?*

Response:

Function	Error	Detected
access()	EINVAL	Yes
	ETXTBSY	No
acos()	EDOM	Yes
asin()	EDOM	Yes
	ERANGE	No
	ERANGE	No

(continued)

Function	Error	Detected
atan()	EDOM	Yes
atan2()	EDOM	Yes
	ERANGE	Yes
catclose()	EBADF	No
	EINTR	No
catgets()	EBADF	No
	EINTR	No
catopen()	EACCES	Yes
	EMFILE	Yes
	ENAMETOOLONG	Yes
	ENFILE	Yes
	ENOENT	Yes
	ENOMEM	Yes
	ENOTDIR	Yes
ceil()	EDOM	Yes
cfsetispeed()	EINVAL	No
cfsetospeed()	EINVAL	No
chmod()	EINVAL	No
chown()	EINVAL	Yes
closedir()	EBADF	Yes
	EINTR	No
cos()	EDOM	Yes
	ERANGE	No
cosh()	EDOM	Yes
erf()	EDOM	Yes
	ERANGE	No
erfc()	EDOM	Yes
	ERANGE	No
exec()	ENOMEM	Yes
	ETXTBSY	No

(continued)

Function	Error	Detected
exp()	EDOM	No
	ERANGE	Yes
fabs()	EDOM	Yes
	ERANGE	No
fcntl()	EDEADLK	Yes
fdopen()	EBADF	No
	EINVAL	No
	EMFILE	Yes
	ENOMEM	Yes
fgetc()	ENOMEM	Yes
	ENXIO	Yes
fgetwc()	ENOMEM	Yes
	ENXIO	Yes
	EILSEQ	Yes
fileno()	EBADF	No
floor()	EDOM	Yes
fmod()	EDOM	Yes
	ERANGE	No
fopen()	EINVAL	No
	EMFILE	Yes
	ENOMEN	Yes
	ETXTBSY	No
fork()	ENOMEM	Yes
fpathconf()	EBADF	Yes
	EINVAL	Yes
fprintf()	EINVAL	Yes
	EILSEQ	Yes
fputc()	ENOMEM	Yes
	ENXIO	Yes

(continued)

Function	Error	Detected
fputwc()	ENOMEM	Yes
	ENXIO	Yes
	EILSEQ	Yes
freopen()	EINVAL	No
	ENOMEM	No
	ETXTBSY	No
frexp()	EDOM	Yes
fseek()	EINVAL	Yes
	EPIPE	No
ftw()	EINVAL	No
getcwd()	EACCES	Yes
	ENOMEM	Yes
getgrgid()	EIO	Yes
	EINTR	No
	EMFILE	Yes
	ENFILE	Yes
getgrnam()	EIO	Yes
	EINTR	No
	EMFILE	Yes
	ENFILE	Yes
getlogin()	EMFILE	Yes
	ENFILE	Yes
	ENXIO	Yes
getpass()	EINTR	No
	EIO	No
	EMFILE	Yes
	ENFILE	No
	ENXIO	No
getpwnam()	EIO	Yes
	EINTR	No
	EMFILE	Yes
	ENFILE	Yes

(continued)

Function	Error	Detected
getpwuid()	EIO	Yes
	EINTR	Yes
	EMFILE	Yes
hsearch()	ENOMEM	Yes
hypot()	EDOM	Yes
	ERANGE	No
iconv()	EBADF	Yes
iconv_close()	EBADF	Yes
iconv_open()	EMFILE	Yes
	ENFILE	Yes
	ENOMEM	Yes
	EINVAL	Yes
isatty()	EBADF	No
	ENOTTY	No
j0()	EDOM	No
	ERANGE	Yes
j1()	EDOM	No
	ERANGE	Yes
jn()	EDOM	No
	ERANGE	Yes
ldexp()	EDOM	Yes
	ERANGE	No
lgamma()	EDOM	No
	ERANGE	Yes
link()	EPERM	Yes
	EXDEV	Yes
log()	EDOM	Yes
	ERANGE	No
log10()	EDOM	Yes
	ERANGE	No

(continued)

Function	Error	Detected
mblen()	EILSEQ	Yes
mbstowcs()	EILSEQ	Yes
mbtowc()	EILSEQ	Yes
modf()	EDOM	Yes
	ERANGE	No
open()	EINVAL	No
	ETXTBSY	No
opendir()	EMFILE	Yes
	ENFILE	Yes
pathconf()	EACCES	Yes
	EINVAL	Yes
	ENAMETOOLONG	Yes
	ENOENT	Yes
	ENOTDIR	Yes
pow()	EDOM	Yes
	ERANGE	No
putenv()	ENOMEM	Yes
read()	ENXIO	Yes
readdir()	EBADF	Yes
rename()	ETXTBSY	No
setvbuf()	EBADF	No
sigaction()	EINVAL	Yes
sigaddset()	EINVAL	Yes
sigdelset()	EINVAL	Yes
sigismember()	EINVAL	Yes
signal()	EINVAL	Yes
sin()	EDOM	No
	ERANGE	Yes

(continued)

Function	Error	Detected
sinh()	EDOM	No
	ERANGE	Yes
sqrt()	EDOM	Yes
strcoll()	EINVAL	No
strerror()	EINVAL	No
strxfrm()	EINVAL	No
tan()	EDOM	Yes
	ERANGE	No
tanh()	EDOM	Yes
	ERANGE	No
tcdrain()	EIO	Yes
tcflush()	EIO	Yes
tcsendbreak()	EIO	Yes
tcsetattr()	EIO	Yes
tmpfile()	EMFILE	Yes
	ENOMEM	Yes
ttyname()	EBADF	Yes
	ENOTTY	Yes
ungetwc()	EILSEQ	Yes
unlink()	ETXTBSY	No
wcscoll()	EINVAL	Yes
wcstombs()	EILSEQ	Yes
wcsxfrm()	EINVAL	Yes
write()	ENXIO	Yes
y0()	EDOM	Yes
	ERANGE	No
y1()	EDOM	Yes
	ERANGE	No

(continued)

Function	Error	Detected
yn()	EDOM	Yes
	ERANGE	No

Rationale

Each of the above error conditions is marked as optional in System Interfaces and Headers, Issue 4 and an implementation may return this error in the circumstances specified or may not provide the error indication.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Section 2.3, *Error Numbers*.

1.1.5 Mathematical Interfaces

Question 9: *What format of floating-point numbers is supported by this implementation?*

Response:

IEEE floating point format.

Rationale

Most implementations support IEEE floating point format either in hardware or software. Some implementations support other formats with different exponent and mantissa accuracy. These differences need to be defined.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Section 1.6, *Relationship to Formal Standards.*

1.1.6 Data Encryption

Question 10: *Are the optional data encryption interfaces provided?*

Response:

Function	Provided
crypt()	Yes

(continued)

Function	Error	Detected
encrypt()	Yes (Decryption capabilities not provided to areas restricted by U.S. export law.)	
setkey()	Yes	

Rationale

Normally, an implementation will either provide all three of these routines or will provide none of them at all. If the routines are not provided, then the implementation must provide a dummy interface which always raises an ENOSYS error condition.

It is also possible that the implementation of the *encrypt()* function may be affected by export restrictions, in which case, the restrictions should be documented here.

For example, historical implementations have supplied all three of these routines outside the U.S.A., but due to export restrictions on the decoding algorithm, a dummy version of *encrypt()* is provided that does encoding but no decoding.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Section 1.2, *Conformance*.

Section 1.2 Process Handling

1.2.1 Process Generation

Question 11: *Which file types (regular, directory, FIFO, special, and so on) are considered to be executable?*

Response:

Only regular file types may be executed.

Rationale

The [EACCES] error associated with `exec` functions occurs in circumstances when the implementation does not support execution of files of the type specified. A list of these file types needs to be provided.

Reference
X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 3, *Functions*, `exec`.

Section 1.3 File Handling

1.3.1 Access Control

Question 12: *What file access control mechanisms does the implementation provide?*

Response:
There is no additional or optional file access control mechanism.

Rationale
System Interfaces and Headers, Issue 4 notes that implementations may provide *additional* or *alternate* file access control mechanisms, or both.

Reference
X/Open CAE Specification, System Interfaces Definitions, Issue 4, Chapter 2, *Glossary*, file access permissions.

1.3.2 Files and Directories

Question 13: *Are any additional or alternate file access control mechanisms implemented that could cause fstat() or stat() to fail?*

Response:
There is no additional or optional file access control mechanism.

Rationale
System Interfaces and Headers, Issue 4 notes that there could be an interaction between additional and alternate access controls and the success of `fstat()` and `stat()`. This would suggest that an implementation can allow access to a file but not allow the process to gain information about the status of the file.

Reference
X/Open CAE Specification, System Interfaces Definitions, Issue 4, Chapter 3, *Functions*, *fstat()* and *stat()*.

1.3.3 Formatting Interfaces

Question 14: *Does the* printf() *function produce character string representations for Infinity and NaN to represent the respective values?*

Response: Yes

Rationale

This behavior is often provided on systems with mathematical functions that produce these results.

Reference

X/Open CAE Specification, System Interfaces and Headers, Issue 4, Chapter 3, *Functions*, fprintf() .

Section 1.4: Internationalized System Interfaces

1.4.1 Coded Character Sets

Question 15: *What coded character sets are supported by the implementation?*

Response:

Solaris 2.4 supports the following coded character sets:

ASCII
ISO 8859-1
JIS X0201
JIS X0208
KS C 5601-87
GB 2312-80
CNS 11643-1986

Rationale

System Interface Definitions, Issue 4 states that conforming implementations support one or more coded character sets, and that each of these includes the portable character set.

Reference

X/Open CAE Specification, System Interfaces Definitions, Issue 4, Chapter 4, *Character Set*.

Question 16: *What is the implementation's underlying internal codeset?*

Response:

ISO 8859-1:1987

Rationale

It is useful to be aware of the underlying codeset of the implementation.

Reference

X/Open CAE Specification, System Interfaces Definitions, Issue 4, Chapter 4, *Character Set*.

1. Commands and Utilities

PRODUCT IDENTIFICATION

Product Identification	Solaris
Version/Release No.	2.5

If you do not supply this component yourself, please identify below the supplier you reference.

INDICATOR OF COMPLIANCE

VSX Test Suite Release	VSC 4.1.4
Testing Agency Name	SunSoft, A Sun Microsystems, Inc. Business
Address	2550 Garcia Avenue Mountain View CA 94043

ENVIRONMENT SPECIFICATION

Enter below details of the hardware and software environment in which testing took place, including compilation routines and installation procedures (if any). Sufficient detail must be supplied to enable conformant behavior and any test results to be reproduced.

SPARC

SPARC running Solaris 2.5. Installation procedures are provided in SPARC: Installing Solaris Software.

To reproduce the test environment, follow these steps:

1. Edit the `/etc/saf/zsmon/_pmtab` file to turn off the ttysoftcarrier detect:

Change the `ttya` and `ttyb` fields from `:y:` to `:n:`. (The colons (:) act as field separators.)

2. Verify that the ttymodes settings in the `/kernel/drv/options.conf` file are set to:

```
2502:1805:bd:8a3b:3:1c:7f:15:4:0:0:0:11:13:1a:19:12:f:17:16
```

3. Disable `ypbind` to allow rebooting of the system:

   a. `cd /usr/lib/netsvc/yp`

   b. `mv ypbind ypbind-`

4. Set the `eeprom` variables that affect the `tty`:

   a. On the keyboard, hit `STOP-A` to display the `prom` prompt.

   b. At the prompt, execute the following steps:

```
setenv ttya-ignore-cd false
setenv ttyb-ignore-cd false
setenv ttya-rts-dtr-off false
setenv ttyb-rts-dtr-off false
```

5. Reboot the system

Note – When installing Solaris, set the time zone by selecting a time zone format that conforms to the POSIX.1 format for TZ defined on page 152 and page 153 of the IEEE Std. 1003.1–1990.

Note – The following option must be added to any compiler line: -D_XOPEN_VERSION=4

Note – The following must be added to any link/load line: /usr/ccs/lib/values_xpg4.o

---

x86

x86 running Solaris 2.5. Installation procedures are provided in x86: Installing Solaris Software.

To reproduce the test environment, follow these steps:

1. Become `root`.

2. Ensure the correct serial port links:

- `/dev/ttya` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/term/a` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/tty00` should be a link to `/devices/isa/asy@3f8,0:a`
- `/dev/ttyb` should be a link to `/devices/isa/asy@2f8,0:a`
- `/dev/term/b` should be a link to `/devices/isa/asy@2f8,0:a`
- `/dev/tty01` should be a link to `/devices/isa/asy@2f8,0:b`

   a. If the `/dev/tty01` link is missing, perform the following:

- Edit `/kernel/drv/asy.conf` and uncomment the COM2 entry
- `# touch /reconfigure`

3. Set the correct serial port permissions:

- `# chmod 666 /devices/eisa/asy*`

4. Turn off the `ttysoftcarrier` detect:
   Using an editor such as `vi`, in the `/etc/saf/zsmon/_pmtab` file, change the next to last field for both the `ttya` entry and the `ttyb` entry from `y` to `n` (the colon (`:`) acts as the field separator):

- `# vi /etc/saf/zsmon/_pmtab`

5. Reboot the system.

Note – When installing Solaris, set the time zone by selecting a time zone format that conforms to the POSIX.1 format for TZ defined on page 152 and page 153 of the IEEE Std. 1003.1–1990.

Note – The following option must be added to any compiler line:
-D_XOPEN_VERSION=4

Note – The following must be added to any link/load line:
/usr/ccs/lib/values_xpg4.o

TEMPORARY WAIVERS

List below references to any temporary waivers granted by X/Open regarding minor errors in the product referenced above. This should include the X/Open reference and the waiver expiration date. The waivers as granted shall be made available with this document on request.

There are no temporary waivers.

1.1 General Attributes

1.1.1 XPG4 Feature Groups

Question 1: *Which of the following Feature Groups are supported by the implementation?*

XCU4 DEVELOPMENT Option

XCU4 FORTRAN Option

Answer: XCU4 DEVELOPMENT Option

Rationale

Commands and Utilities, Issue 4 states that the system may provide one or more of the Feature Groups listed.

Reference

X/Open CAE Specification, Commands and Utilities, Issue 4, Section 1.2, Conformance and Section 1.3, Status of Interfaces.

1.1.2 XPG4 Development Utilities

Question 2: *Does the implementation support the development utility dis?*

Answer: Yes

Rationale

Commands and Utilities, Issue 4 states that optional utilities listed in Section 1.3.1 on page 2 need not be provided. Section 1.3.1 lists the dis utility as OPTIONAL.

Reference

X/Open CAE Specification, Commands and Utilities, Issue 4, Section 1.2, Conformance and Section 1.3, Status of Interfaces.

1.1.3 XPG4 Unsupportable Utilities

Question 3: *Are the following utilities supported by the implementation?*

Answer:

Utility	Supported
cancel	YES
cu	YES
lpstat	YES
uucp	YES
uulog	YES
uuname	YES
uupick	YES
uuto	YES

Rationale

Commands and Utilities, Issue 4 states that the system need not provide the possibly unsupportable utilities listed.

Reference

X/Open CAE Specification, Commands and Utilities, Issue 4, Section 1.2, Conformance and Section 1.3, Status of Interfaces.

Question 4: *Are the following options to the specified utilities supported by the implementation?*

Answer:

Options	Supported
-s option to *ar*	YES
-E option to *cc*	YES
-P option to *cc*	YES
-S option to *cc*	YES

(continued)

Options	Supported
-m option to *lp*	YES
-o option to *lp*	YES
-t option to *lp*	YES
-w option to *lp*	YES
-p option to *mail*	YES
-q option to *mail*	YES
-r option to *mail*	YES
-t option to *mail*	YES
-n option to *pg*	YES
-z option to *sort*	NO (See Note)
-s option to *ar*	YES
-r option to *sum*	YES
+m[n] option to *tabs*	YES
-q option to *uustat*	YES
-r option to *uustat*	YES
-j option to *uux*	YES

The support of these options to the specified utilities are an extension to the X/Open CAE Specification standard. These options need not be supported on all XSI-conformant systems.

Rationale

Possibly unsupportable feature.

Reference

X/Open CAE Specification, Commands and Utilities, Issue 4.

Note – *sort* ignores the -z option and allocates as much memory as needed to sort the lines. It will not fail if a line longer than `recsz` is encountered.

1.1.4 POSIX.2 Supported Features

Question 5: *Are the following features available on the system?*

Answer:

Macro Name	Meaning	Provided
POSIX2_C_DEV	The implementation supports the C-language development utilities in Annex A of POSIX.2.	Yes
POSIX2_CHAR_TERM	The implementation supports character-based terminals.	Yes
POSIX2_FORT_DEV	The implementation supports the FORTRAN language development utilities in Annex C of POSIX.2.	No
POSIX2_FORT_RUN	The implementation supports the FORTRAN runtime utilities in Annex C of POSIX.2.	No
POSIX2_SW_DEV	The implementation supports the software development utilities in Section 6 of POSIX.2.	Yes
POSIX2_UPE	The implementation supports the User Portability Utilities in Section 5 of POSIX.2.	Yes

Rationale

POSIX.2 states that a POSIX.2 conforming implementation may provide one or more of the following: the User Portability Option, the Software Development Utilities Option, the C-Language Development Utilities Option, the FORTRAN Development Utilities Option, or the FORTRAN Runtime Utilities Option. It also lists POSIX2_CHAR_TERM and POSIX2_LOCALEDEF as the implementation options.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 1.3.

Question 6: *Which terminal types are supported by the implementation?*

Answer: Any terminal for which a user can obtain a TERMINFO description and run through the `tic` utility.

Rationale

The implementation shall document which terminal types it supports and which of the features and utilities are not supported by each terminal.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 2.14.

Question 7: *Does the implementation support moving files across file systems using the mv utility?*

Answer: Yes

Rationale

The mv utility shall perform actions equivalent to the POSIX.1 *rename()* function, called with the following arguments:
(a) The source_file operand is used as the old argument.
(b) The destination path is used as the new argument.

If the links named by new and old are on different file systems, and the implementation does not support links between file systems, the *rename()* function shall return -1 and set errno to [EXDEV].

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 2.14.

POSIX–PART 1: SYSTEM APPLICATION PROGRAM INTERFACE, Section 5.5.3

Question 8: *Does the implementation require write permission for an existing directory in order for mv utility to rename that directory?*

Answer: No

Rationale

The mv utility shall perform actions equivalent to the POSIX.1 *rename()* function, called with the following arguments:

(a) The source_file operand is used as the old argument.

(b) The destination path is used as the new argument.

If write permission is required and is denied for a directory pointed to by the old or new arguments, the *rename()* function shall return -1 and set errno to [EACCES].

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 4.43.

POSIX–PART 1: SYSTEM APPLICATION PROGRAM INTERFACE, Section 5.5.3.

Question 9: *Does the implementation consider it to be an error if a nonprivileged process attempts to rename a file or directory using the mv utility while that file or directory is being used by the system or another process?*

Answer: No

Rationale

The mv utility shall perform actions equivalent to the POSIX.1 *rename()* function, called with the following arguments:

(a) The source_file operand is used as the old argument.

(b) The destination path is used as the new argument.

If the directory named by old or new cannot be renamed because it is being used by the system or another process and the implementation considers this to be an error, the *rename()* function shall return -1 and set errno to [EBUSY].

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 4.43.

POSIX–PART 1: SYSTEM APPLICATION PROGRAM INTERFACE, Section 5.5.3.

Question 10: *Does the implementation support creating hard links across file systems using the ln utility?*

Answer: No

Rationale

The ln utility shall perform actions equivalent to the POSIX.1 *link()* function using source_file as the path1 argument, and the destination path as the path2 argument.

If the link named by new and the file named by existing are on different file systems, and the implementation does not support links between file systems, the *link()* function shall return -1 and set errno to [EXDEV].

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 4.33.

POSIX–PART 1: SYSTEM APPLICATION PROGRAM INTERFACE, Section 5.3.4.

Question 11: *Does the implementation support the floating point conversions in the format string of the printf utility?*

Answer: Yes

Rationale

The e, E, f, g, and G conversion specifications need not be supported in the format string of the format operand of the printf utility.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 4.50.

Question 12: *In the system's handling of Basic Regular Expressions, do the following characters in a subexpression anchor that subexpression?*

Answer:

Feature	Response
The circumflex (^) character	NO
The dollar-sign ($) character	NO

Rationale

The implementation may treat circumflex as an anchor when used as the first character of a subexpression. The implementation may treat a dollar-sign as an anchor when used as the last character of a subexpression.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 2.8.

Question 13: *Does the implementation support the use of numeric signal numbers in the shell's trap command?*

Answer: Yes

Rationale

An implementation may allow numeric signal numbers for the conditions in the trap built-in utility as an extension.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 3.14.13.

1.1.5 POSIX.1 Supported Features

Question 14: *Does the implementation support the POSIX.1 C language interfaces?*

Answer: Yes

Rationale

If the implementation conforms to POSIX.1, all but the shell special built-in utilities can be *exec()*'ed.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 2.3.

1.1.6 Implementation-Dependent Limits Affecting Utilities

Question 15: *What are the minimum values associated with the following symbols?*

Answer:

Macro Name	Meaning	Min. Value
BC_BASE_MAX	Maximum *obase* value allowed by the *bc* utility.	99
BC_DIM_MAX	Maximum number of elements permitted in an array by the *bc* utility.	2048
BC_SCALE_MAX	Maximum scale value allowed by the *bc* utility	99
BC_STRING_MAX	Maximum length of a string constant accepted by the *bc* utility.	1000

(Continued)

Macro Name	Meaning	Min. Value
COLL_WEIGHTS_MAX	Maximum number of weights that can be assigned to an entry of the LC_COLLATE order keyword in the locale definition file.	2
EXPR_NEST_MAX	Maximum number of expressions that can be nested within parentheses by the expr utility.	32
LINE_MAX	Maximum length in bytes including the trailing newline of a utility's input line when the utility is described as processing text files.	2048
RE_DUP_MAX	Maximum number of repeated occurrences of a regular expression permitted when using interval notation.	255

Rationale

The minimum value of these symbols on any XSI conforming system must not be more restrictive than the value of the corresponding POSIX2_ symbols. A specific conforming implementation may provide a higher minimum value than this.

Reference

POSIX–PART 2: SHELL AND UTILITIES, Section 2.13.

Section 1.2: Internationalized System Interfaces

1.2.1 Coded Character Sets

Question 16: *What coded character sets are supported by the implementation?*

Answer: Any character set that can be encoded in EUC: ISO 8859-1, CNS 11643-1992, JIS X0208, KS C 5861-1992, GB23/2-80

Rationale

System Interface Definitions, Issue 4 states that conforming implementations support one or more coded character sets, and that each of these includes the portable character set.

Reference
X/Open CAE Specification, System Interfaces Definitions, Issue 4, Chapter 4, Character Set

Question 17: *What is the implementation's underlying internal codeset?*

Answer: Encoding method uses EUC (Extended UNIX codeset).

Rationale
It is useful to be aware of the underlying codeset of the implementation.

Reference
X/Open CAE Specification, System Interface Definitions, Issue 4, Chapter 4, Character Set.

6