X/Open and XPG3 5

The X/Open consortium was established to make multivendor open systems a practical reality. X/Open takes existing standard interfaces and adapts them to the specifics of open systems. These interfaces comprise what is known as the Common Applications Environment (CAE) and are documented in the X/Open Portability Guide.

This chapter discusses the compliance of Solaris 2.5 to the programming interface specifications detailed in the X/Open Portability Guide, Issue 3 (XPG3).

The X/Open Portability Guide, Issue 3

In 1988, X/Open published the X/Open Portability Guide Issue 3, commonly referred to as "XPG3." It is a collection of seven volumes that includes the interfaces specified in the IEEE 1003.1-1988 POSIX standard.

Adherence to the programming interface specifications contained in XPG3 ensures application portability at the source code level. Compliance with these interfaces is determined through an extensive set of conformance tests and is assured through the X/Open branding process which entitles a product to bear the X/Open trademark.

5

Note – In 1992, X/Open published Issue 4 of the Portability Guide, (XPG4). It retains compliance to the IEEE 1003.1-1988 standard but is extended to the ISO/IEC updated POSIX.1 standard and the ISO/IEC C language standard. While XPG3 is still available and systems and components can still be branded to XPG3, XPG4 offers significant additional capability. For more information on XPG4, see Chapter 6, "X/Open and XPG4."

This chapter identifies Solaris 2.5 as a conforming implementation of XPG3, and displays the XPG3 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 3.

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.

Components of the Common Applications Environment are categorized into three levels: BASE, PLUS, and OPTIONS. A system that provides all of the BASE components is awarded an XPG3 BASE profile trademark. A system that implements all of the BASE and PLUS components may bear the XPG3 PLUS profile trademark.

Figure 5-1 The XPG3 Base Brand Logo

Solaris has earned the XPG3 BASE brand. Solaris products and software products from independent software vendors that have received XPG3 branding are described below:

- **Window Management (OpenWindows)**—The Solaris window system, which supports the OPEN LOOK Graphical User Interface, has earned the XPG3 brand by implementing the programmer's interface to the X Window System. OpenWindows supports the Window Management component of the X/Open PLUS 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 VSX3 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 VSX4.2.4 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 and carries the XPG3 brand when run on Solaris.
- **Sun Pascal 3.0.1**—Sun's compiler for the Pascal programming language is fully compliant with the ISO standard and carries the XPG3 brand when run on Solaris.
- **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 OPTIONS 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 OPTIONS 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 OPTIONS level.

5

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

Chapter 2: 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.

Conformance Reference

Indicator of Compliance

VSX Test Suite Release	VSX 4.3.2
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, 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.

---

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.

Temporary Waivers

List below references to any temporary waivers granted by X/Open in respect to 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.

Section 2.1 General Attributes

2.1.1 POSIX.1 Supported Features

Question 1: *Which of the following options, specified in the* <**unistd.h**> *header fileSCC are available on the system?*

Answer:

Macro Name	Meaning	Provided
_POSIX_CHOWN_RESTRICTED	The use of **chown()** is restricted	Variable
_POSIX_JOB_CONTROL	Job Control option	Yes
_POSIX_NO_TRUNC	Long pathname components generate an error	Variable
_POSIX_SAVED_IDS	Effective user and group IDs are saved	Yes
_POSIX_VDISABLE	Terminal special characters can be disabled	Variable

When native SunOS file systems and terminal drivers are used, _POSIX_CHOWN_RESTRICTED is supported, _POSIX_NO_TRUNC is supported, and _POSIX_VDISABLE has the value '0', or '\0' in C language source. When other file system types are used, such as through NFS, or terminal drivers from third party vendors, the results may vary and can be queried using `pathconf()` and `fpathconf()`.

Rationale

For an X/Open conforming implementation, the `_POSIX_SAVED_IDS` option must be provided. The other options may or may not be provided. The provision of the file system related options can vary within a system. For example, a system which has traditionally supported both System V and BSD type file systems may provide a mechanism whereby the option is enforced for certain files or processes but not for others. This technique can be used to achieve a degree of backwards compatibility that would not otherwise be possible.

Reference

XPG3 Volume 2, page 579–<**unistd.h**>

2.1.2 C Standard

Question 2: *Does the implementation only support* **Common Usage C** *or also support* **ANSI C Standard** *interface definitions?*

Answer:
Both Common Usage C and ANSI C are provided.

Rationale
The POSIX.1 standard allows for a conforming system to support either Common Usage C or ANSI C Standard interface definitions. The XPG is based on a Common Usage C definition but does not prohibit an ANSI C implementation. A Common Usage C definition must provide function declarations for the C language functions in the XPG as well as providing function semantics that conform to the XPG. An ANSI C Standard interface must provide function prototypes and ANSI C semantics as well as providing XPG semantics. There are no known areas of contradiction between the ANSI C and the XPG semantics.

Reference
XPG3 Volume 2, page 12 - *The Compilation Environment*

2.1.3 Limit Values

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

Answer:

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

MAX_INPUT	Max bytes in a terminal input queue	255	512
NAME_MAX	Max characters in a filename	14	See note
OPEN_MAX	Max number of files open in a process	16	See note
PASS_MAX	Max significant characters in a password	8	8
PATH_MAX	Max characters in a pathname	255	See note
PIPE_BUF	Max bytes in an atomic write to a pipe	512	5120
NGROUPS_MAX	Max number of supplementary group IDs	0	16
TMP_MAX	Max number of unique temporary file names	17576	17576

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 XPG3.

Rationale

Each of these limits can vary within bounds set by the X/Open Portability Guide. The minimum value that a limit can take on any X/Open 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

XPG3 Volume 2, page 538 - **<limits.h>**

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

Answer:

Macro Name	Meaning	Value
CHAR_BIT	Number of bits in a char	8
LONG_BIT	Number of bits in a long	32
WORD_BIT	Number of bits in a word	32
DBL_DIG	Digits of precision of a double	15
DBL_MAX	Maximum decimal value of a double	1.7976931348623157E+308
FLT_DIG	Digits of precision of a float	6
FLT_MAX	Maximum decimal value of a float	3.40282347E+38

Rationale

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

Reference

XPG3 Volume 2, page 537 - **<limits.h>**

2.1.4 Error Conditions

Question 5: *Which of the following optional errors listed in the XPG are detected in the circumstances specified?*

Answer:

Function	Error	Detected
access()	EINVAL†	Yes
	ETXTBSY	No
atof()	ERANGE	Yes
atoi()	ERANGE	Yes

(continued)

Function	Error	Detected
atol()	ERANGE	Yes
cfsetispeed()	EINVAL	No
cfsetospeed()	EINVAL	No
chmod()	EINVAL	No
chown()	EINVAL†	Yes
closedir()	EBADF†	Yes
exec	ENOMEM†	Yes
	ETXTBSY	No
fcntl()	EDEADLK†	Yes
fdopen()	EBADF	No
	EINVAL	No
feof()	EBADF	No
ferror()	EBADF	No
fileno()	EBADF	No
fopen()	EINVAL	No
	ETXTBSY	No
freopen()	EINVAL	No
	ETXTBSY	No
fork()	ENOMEM	Yes
fseek()	EINVAL	Yes
ftw()	EINVAL	No
getcwd()	EACCES†	Yes
isatty()	EBADF	No
	ENOTTY	No

(continued)

Function	Error	Detected
open()	EINVAL	No
	ETXTBSY	No
opendir()	EMFILE†	Yes
	ENFILE†	Yes
pathconf()	EACCES†	Yes
	EINVAL†	Yes
	ENAMETOOLONG†	Yes
	ENOENT†	Yes
	ENOTDIR†	Yes
fpathconf()	EBADF†	Yes
	EINVAL†	Yes
printf()	EINVAL	No
readdir()	EBADF†	Yes
rename()	ETXTBSY	No
scanf()	EINVAL	No
setvbuf()	EBADF	No
sigaddset()	EINVAL†	Yes
sigdelset()	EINVAL†	Yes
sigismember()	EINVAL†	Yes
strcoll()	EINVAL	No
strerror()	EINVAL	No
strtol()	EINVAL	Yes
	ERANGE	Yes
strxfrm()	EINVAL	No
unlink()	ETXTBSY	No

Rationale

Each of the above error conditions is marked as optional in the XPG and an implementation may return this error in the circumstances specified or may not provide the error indication. Those items marked with a † are also considered to be optional error conditions in POSIX.1. The EINVAL error condition for the three functions **sigaddset()**, **sigdelset()**, and **sigismember()** are mandated in the XPG but are considered optional in POSIX.1. An X/Open conforming implementation will always produce these errors, but a POSIX.1 conforming implementation may not.

Reference

XPG3 Volume 2, page 32– Error Numbers.

2.1.5 Mathematical Interfaces

Question 6: *What format of floating point numbers is supported by this implementation?*

Answer:

IEEE floating point format is supported.

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.

Question 7: *Is* **long double** *form supported and what precision is associated with this form?*

Answer:

Long double uses 16 bytes. The low order 112 bits are used to hold the mantissa, the next 15 bits hold the exponent, and the high order bit is used as the sign bit.

Rationale

The long double format can vary both in length and precision. If it is supported, other than as a synonym for double, the format needs to be described.

Reference

XPG3 Volume 2, page 328 - printf()
XPG3 Volume 2, page 362 - scanf()

2.1.6 Data Encryption

Question 8: *Are the optional data encryption interfaces provided?*

Answer:

crypt()	Yes
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 the routines outside the USA, but due to export restrictions on the decoding algorithm, a dummy version of **encrypt()** is provided that does encoding, but not decoding.

Reference

XPG3 Volume 2, page 3 - *Status of Interfaces*

Section 2.2 Process Handling

2.2.1 Process Generation

Question 9: *Which file types (regular, directory, FIFO, special etc.) are considered to be executable?*

Answer:

Only regular files 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
XPG3 Volume 2, page 129 -exec

2.2.2 Process Termination

Question 10: *Is the* SIGCHLD *signal sent to the parent process when a child exits?*

Answer:
Yes

Rationale
Some systems support the sending of SIGCHLD in these circumstances. This is mandatory if job control is supported.

Reference
XPG3 Volume 2, page 132 -exit()

2.2.3 Process Environment

Question 11: *Is the* **setpgid()** *interface provided?*

Answer:
Yes

Rationale
This interface is mandatory on systems which support job control and may be provided on other systems.

Reference
XPG3 Volume 2, page 3 - *Status of Interfaces*

Section 2.3 File Handling

2.3.1 Access Control

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

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

Rationale
The XPG (and POSIX) allow an implementation to provide either additional or alternate file access control mechanisms other than the standard access control mechanism. The document should either describe or provide a reference to the details of alternate or additional access mechanisms. In particular, the method

by which an application can execute (using standard file access control) should be explained, and details of the changes required to utilize alternate or additional access mechanisms should be given.

Reference
XPG3 Volume 2, page 16 - *File Access Permissions*

2.3.2 Files and Directories

Question 13: *Are any extended security controls implemented that could cause* fstat() *or* stat() *to fail?*

Answer:
No

Rationale
The XPG notes that there could be an interaction between extended security 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
XPG3 Volume 2, page 478 -tempnam()

2.3.3 Formatting Interfaces

Question 14: *Is the L modifier to* **printf()** *and* **scanf()** *supported in this implementation?*

Answer:
Yes

Rationale
The XPG notes that the L modifier, which is exactly equivalent to the l modifier when the implementation does not differentiate between double and long double, is not supported on all systems and is only included for compatibility with ANSI C.

Reference
XPG3 Volume 2, page 328 - printf()
XPG3 Volume 2, page 362 - scanf()

Question 15: *Does the* **printf()** *function produce character string representations for Infinity and NaN to represent the respective special double precision values?*

Answer:
Yes

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

Reference
XPG3 Volume 2, page 331 - printf()

Section 2.4 General Terminal Interface

2.4.1 Interfaces Supported

Question 16: *Are the following terminal control interfaces provided?*

tcgetpgrp() tcsetpgrp()

Answer:
Yes

Rationale
These interfaces are mandatory for implementations that support **job control**. Implementations that do not support **job control** may either always return the error indication [ENOSYS] or may provide the interface with the behavior specified for an implementation that supports **job control**. The latter case is useful for implementations that support only part of the **job control** specifications.

Reference
XPG3 Volume 2, page 471 - tcgetpgrp
XPG3 Volume 2, page 475 - tcsetpgrp

Section 2.5 Internationalized System Interfaces

2.5.1 Codesets

Question 17: *Does the implementation support the* **ISO 8859-1:1987** *codeset for data transmission?*

Answer:
Yes

Rationale

The XPG defines the ISO 8859-1:1987 as the major Western European transmission codeset and also recommends its use as the corresponding internal codeset.

Reference

XPG3 Volume 3, page 19 - *Character Codesets and Text Transfer*

Question 18: *Does the implementation use the* **ISO 8859-1:1987** *as its internal codeset?*

Answer:

Yes

Rationale

The XPG defines the ISO 8859-1:1987 as the major Western European transmission codeset and also recommends its use as the corresponding internal codeset.

Reference

XPG3 Volume 3, page 19 - *Character Codesets and Text Transfer*

2.5.2 Regular Expression Interfaces

Question 19: *What form of regular expression syntax is supported by the* **regexp()** *interface?*

Answer:

Simple regular expression

Rationale

The **regexp()** interface may support either the simple regular expression or the simple internationalized regular expression syntax as defined in the XPG3 Volume 3 - *Supplementary Definitions*.

Reference

XPG3 Volume 3, pages 49-51 - *Regular Expressions*

Chapter 3: 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.

Conformance Reference

Indicator of Compliance

None

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 and x86, running Solaris. Installation procedures are provided in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

Conformance Expectation

Volume 1 of XPG3 recognizes that convergence of implementations towards a common specification for commands and utilities is not yet complete and therefore does not require a vendor to supply all of the commands and utilities (and individual options) specified in XPG3.

This chapter explicitly identifies those commands and utilities not supplied by the vendor and any supplied that do not conform to the published specification. (Reference: XPG3 Volume 1, page 1.)

Section 3.1 Basic Utilities

3.1.1 Supported Commands

Question 1: *Which of the basic utilities (non-development utilities) defined in the XPG are not provided with the implementation?*

Answer:
All the defined utilities are provided.

Rationale
The XPG Volume 1 states that "this volume in its current form is useful only as a guide to portability, but it is not possible to precisely define or test conformance to it." This question determines whether or not the implementation provides a command of the name specified in the XPG; it does not attempt to determine whether it supports the semantics of that command. The (optional) development utilities are excluded from this question and are dealt with in the next section of the questionnaire.

Reference
XPG3 Volume 1, page 1 - *Introduction*

3.1.2 Command Behavior

Question 2: *In what ways do the commands provided by the implementation behave differently from the specifications contained in the XPG?*

Answer:
The -n option to ps is not supported.

Rationale
This question provides a greater degree of granularity than the previous question, requiring the semantic differences associated with the commands to be specified. Again, the question relates to the basic utilities rather than the development utilities. The question only relates to the semantics of the options specified within the XPG; implementation specific extensions should not be documented.

Section 3.2 Development Utilities

3.2.1 Supported Commands

Question 3: *Which of the development utilities defined in the XPG are not provided with the implementation?*

Answer:
The sdb utility is not provided.

Rationale
The XPG Volume 1 states that "The development utilities might not be present in all X/Open compliant systems; in designated (**DEVELOPMENT**) systems all of the development utilities must be present and must conform to the published definition."

Reference
XPG3 Volume 1, page 2 - *Status of Interfaces*

3.2.2 Command behavior

Question 4: *In what ways do the development utilities provided by the implementation behave differently from the specifications contained in the XPG?*

Answer:
The make utility looks for sccs s.files in the directory ./SCCS instead of in the current directory.

Rationale
This question provides a greater degree of granularity than the previous question, requiring the semantic differences associated with the development utilities to be specified. The question only relates to the semantics of the options specified within the XPG; implementation-specific extensions should not be documented.

Section 3.3 Internationalization Option

3.3.1 Commands and Utilities

Question 5: *Is an internationalized environment, reflecting changes in the locale setting as described in* **XPG Volume 1- XSI Commands and Utilities**, *supported*?

Answer:

Command	Behavior Specified in XPG3	Supported
ar	LC_TIME affects date format	Yes
awk	LC_COLLATE, LC_CTYPE affect regular expression matching	No

(continued)

Command	Behavior Specified in XPG3	Supported
	LC_COLLATE affects the behavior of string comparisons	No
	LC_NUMERIC affects the behavior of the radix character	No
comm	LC_COLLATE affects sorting sequence	No
cp,ln,mv	LANG affects yes string	Yes
cpio	LC_COLLATE, LC_CTYPE affect filename pattern matching	No
	LC_TIME affects date format	Yes
date	LC_TIME affects date formatting options	Yes
ed,red	LC_COLLATE, LC_CTYPE affects regular expression matching	No
	LC_CTYPE is used to determine whether characters are printable	Yes
egrep	LC_COLLATE, LC_CTYPE affects regular expression matching	No
	LC_CTYPE is used to determine character classification (alphabetic, upper case, lower case)	Yes
expr	LC_COLLATE, LC_CTYPE affects regular expression matching	No
	LC_COLLATE affects the behavior of relational operators	No
fgrep	LC_CTYPE is used to determine character classification (alphabetic, upper-case, lower case)	Yes
find	LANG affects yes string	No
	LC_COLLATE, LC_CTYPE affects filename pattern matching	No

(continued)

Command	behavior Specified in XPG3	Supported
grep	LC_COLLATE, LC_CTYPE affects regular expression matching	No
	LC_CTYPE is used to determine character classification (alphabetic, upper-case, lower case)	Yes
join	LC_COLLATE affects sorting sequence	No
lpstat	LC_TIME affects date format	Yes
ls	LC_COLLATE affects sorting sequence	Yes
	LC_CTYPE is used to determine whether a character is printable	Yes
	LC_TIME affects date format	Yes
mail	LC_TIME affects date format	Yes
mailx	LC_COLLATE, LC_CTYPE affects filename pattern matching	No
	LC_TIME affects date format	Yes
pg	LC_COLLATE, LC_CTYPE affects filename pattern matching	No
pr	LC_TIME affects date format	Yes
	LC_CTYPE is used to determine whether a character is printable	Yes
ps	LC_TIME affects date format	Yes
`rm,rmdir`	`LANG` affects yes string	No
`sed`	`LC_COLLATE`, `LC_CTYPE` affects regular expression matching	No
	`LC_CTYPE` is used to determine whether a character is printable	Yes
`sh`	`LC_COLLATE`, `LC_CTYPE` affects filename pattern matching	No

(continued)

Command	behavior Specified in XPG3	Supported
	LC_CTYPE is used to determine whether a character is alphabetic	No
sort	LC_COLLATE affects sorting sequence	No
	LC_CTYPE affects character classification (alphabetic, upper case, printing)	Yes
	LC_NUMERIC affects the determination of the radix character	No
tar	LC_TIME affects date format	No
	LANG affects yes string	No
tr	LC_COLLATE, LC_CTYPE affects bracketed expressions	No
	LC_CTYPE affects the definition of the character universe	No
uniq	LC_COLLATE affects sorting sequence	No
uucp	LC_TIME affects date format	No
uustat	LC_TIME affects date format	No
wc	LC_CTYPE is used to determine white-space characters	Yes
who	LC_TIME affects date format	Yes
yacc	LC_CTYPE is used to determine character classification	Yes

Rationale

This behavior is collectively optional, that is, it should be provided for all commands listed (subject to sections 3.1 and 3.2, which identify those commands not supplied by the vendor and those which do not fully support the X/Open specification).

Reference

XPG3 Volume 1, pages 4-5 - *Status of Interfaces*.

3.3.2 Regular Expressions in Commands

Question 6: *Which form of regular expression syntax is supported by those commands which use regular expressions?*

Answer:

Command	Regular Expression Syntax Supported
awk	Extended
csplit	Simple
ed	Simple
egrep	Extended
ex	Simple
expr	Simple
grep	Simple
lex	Extended
pg	Simple
sdb	sdb is not supported
sed	Simple
vi	Simple

Rationale

The XPG Volume 3 - *XSI Supplementary Definitions* requires that an internationalized set of commands will provide regular expression syntax for the above commands in one of the forms specified for that command. The XPG encourages the implementation of internationalized regular expressions for all of the above utilities. It should be noted that the `sdb` command is an optional development utility and may not be available on all XPG conforming systems.

Reference

XPG3 Volume 3, pages 49-51 - *Regular Expressions*

Chapter 4: C Language

SPARC

Product Identification

Product Identification	SPARCompiler C
Version/Release No.	2.0.1

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

Conformance Reference

Indicator of Compliance

VSX Test Suite Release	VSX 4.3.2
Testing Agency Name	Mindcraft, Inc.
Address	410 Cambridge Avenue Palo Alto, California 94306

x86

Product Identification

Product Identification	ProCompiler C
Version/Release No.	2.0.1

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

Conformance Reference

Indicator of Compliance

VSX Test Suite Release	VSX 4.3.2
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 and x86, running Solaris. Installation procedures are provided in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

Temporary Waivers

List below references to any temporary waivers granted by X/Open in respect to 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.

Section 4.1 Implementation Limits

Question 1: *What limits does the implementation impose on the significant part of an identifier?*

Answer:

External identifiers	No limits; all characters are significant
Non-External identifiers	No limits; all characters are significant

Rationale

The XPG states that, while there is no limit to the length of an identifier, only a certain number of characters are significant. The XPG points out that there must be at least eight characters for a non-external name, but may be less for external names.

Reference

XPG 3 Volume 4, page 3 - *Lexical Conventions*

4.2 General

Question 2: *What truncation rules are applied when a floating value is converted to an integral value?*

Answer:

Truncation of floating point values is always towards zero.

Rationale

The XPG states that such conversions are machine dependent. In particular, the XPG points out the differences related to the truncation of negative numbers.

Reference

XPG Volume 4, page 10 - *Conversions*

Question 3: *What truncation rules are applied when using the division operator and either of the operands is negative?*

Answer:

Truncation towards zero

Rationale:

The XPG states that such truncations are machine dependent.

Reference

XPG Volume 4, page 16 - *Expressions*

Chapter 11: Terminal Interfaces

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.

Conformance Reference

Indicator of Compliance

None

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 and x86, running Solaris. Installation procedures are given in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

Temporary Waivers

List below references to any temporary waivers granted by X/Open in respect to 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.

Chapter 12: Window Management

Product Identification

Product Identification	OpenWindows
Version/Release No.	3.0 and subsequent releases

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

Conformance Reference

Indicator of Compliance

None

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 and x86, running Solaris. Installation procedures are provided in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

Temporary Waivers

List below references to any temporary waivers granted by X/Open in respect to 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.

Chapter 14: Inter-process Communication

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.

Conformance Reference

Indicator of Compliance

None

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 and x86, running Solaris. Installation procedures are provided in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

Temporary Waivers

List below references to any temporary waivers granted by X/Open in respect to 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.

Chapter 15: Source Code Transfer

Section 15.1 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.

Conformance Reference

Indicator of Compliance

None

5

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 and x86, running Solaris. Installation procedures are provided in *SPARC: Installing Solaris Software or x86: Installing Solaris Software*.

SPARC

For floppy disk hardware:

SPARCstation with external or internal floppy disk drive (part number X554H)

For magnetic tape hardware:

SPARC Office Servers (SPARCsystem 330 or SPARCsystem 470) with 1/2-inch tape drive subsystem (part number 680A), and
SPARC Data Center Servers (SPARCserver 390 or SPARCserver 490) with 1/2-inch tape drive subsystem (part numbers 682A or 683A)

x86

For Source Code Transfer software:

x86, running Solaris 2.5. Installation procedures are given in the *Solaris System Configuration and Installation Guide*.

For floppy disk hardware:

x86 machine, with external or internal floppy disk drive

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.

Formats

Question 1: *Which exchange media format(s) may be written by the system?*

Answer:

80 track floppy disk	Yes
40 track floppy disk	No
1600 bpi PE magnetic tape	Yes

Rationale

XPG3 states that standards are referenced for transfer of floppy disks and magnetic tapes between machines. Because of the different nature of X/Open conformant systems, it is not possible to define a single portable medium which is supported across the whole range of systems.

Reference

XPG3 Volume 3, Chapters 15, 16, and 17

Question 2: *Which exchange media format(s) may be read by the system?*

80 track floppy disk	Yes
40 track floppy disk	No
1600bpi PE magnetic tape	Yes

Rationale

XPG3 states that standards are referenced for transfer of floppy disks and magnetic tapes between machines. Because of the different nature of X/Open conformant systems, it is not possible to define a single portable medium which is supported across the whole range of systems. In addition, some systems can read a wider range of formats than they can write.

Reference

XPG3 Volume 3, Chapters 15, 16, and 17

Utilities

Question 3: *Which utilities are used to create and read the archive formats specified in* **XPG Volume 3-XSI Supplementary Definitions***?*

Answer:

Format	Creating	Reading
Extended `tar`	`cpio -H USTAR`	`cpio -H USTAR`
`cpio`	`cpio -H odc`	`cpio -H odc`

Rationale

There is no explicit definition as to the commands that must be used to create and retrieve these archives. On most systems this will be achieved by the `tar` and `cpio` commands. There are other commands available which produce these archives. On some implementations the command may need a special option to enable reading of the specified formats with the "standard" option being to create archives which are backwards compatible with previous versions of the command.

Reference

XPG3 Volume 3, page 151-2 – Utilities

Invalid Files Names

Question 4: *What file name is used to contain data from the archive in the case that the file name on the archive is invalid for the system on which the file hierarchy is being created?*

Answer:

Format	File Name
Extended `tar`	All legal file names in a USTAR archive are legal in the filesystem.
`cpio`	All legal file names in a cpio archive are legal in the filesystem.

Rationale

Because an archive can contain non-portable file names it is necessary for an archive reading utility to be able to generate a file and store the data associated with a non-portable file name when this is encountered on the archive. There may be a need to generate a number of such file names in the same directory and the specification should detail the algorithm used to generate these file names.

Reference

XPG3 Volume 3, page 151– Utilities

Multivolume Archives

Question 5: *How does the archive reading utility determine which file to read as the next volume when an end-of-media condition is encountered?*

Answer:

Format	Method
Extended `tar`	The `tar` utility prompts the user for the pathname of the next file in the archive. (The path need not name a device.)
`Cpio`	The `cpio` utility prompts the user for the pathname of the next file in the archive. (The path need not name a device.)

Rationale

In many cases the utility will prompt the user for the path name of the device to use for the next volume. There may be extensions to the utility syntax that allow the definition of alternate addresses for subsequent volumes.

Reference

XPG3 Volume 3, page 151-2 – Utilities.

X/Open Specification and Related Publications

The X/Open Portability Guide is published by Prentice-Hall in the United States. The set is comprised of the seven volumes listed below; the ISBN number follows the volume title:

- *Volume 1: XSI Commands and Utilities, 0-13-68555835-X*
- *Volume 2: XSI System Interface and Headers, 0-13-685843-0*

5

- *Volume 3: XSI Supplementary Definitions, 0-13-685850-3*
- *Volume 4: Programming Languages, 0-13-685868-6*
- *Volume 5: Data Management, 0-13-685876-7*
- *Volume 6: Window Management, 0-13-685884-8*
- *Volume 7: Networking Services, 0-13-685892-9*