

SPARC COMPLIANCE DEFINITION 2.3

SCD
2.3

SPARC International

© 1990, 1991, 1992, 1993, 1994, 1995 SPARC International Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owners.

The manual pages for socket functions are

© 1992, 1993 The Regents of the University of California. All rights reserved

Includes material copyrighted by UNIX System Laboratories, Inc., a subsidiary of Novell, Inc. Reprinted with permission.

The SPARC Compliance Definition 2.3 is published and printed by SPARC International.

Any comments relating to the material contained herein may be submitted to:

SPARC International Inc.

535 Middlefield Road, Suite 210

Menlo Park, California 94025

ATTN: Ghassan Abbas (abbas@sparc.com)

Trademarks

SPARC® is a registered trademark of SPARC International, Inc.

SPARCstation™ is a trademark of SPARC International, Inc.

Products bearing SPARC® trademarks are based on an architecture developed by Sun Microsystems, Inc.

ONC™ and SunOS™ are trademarks of Sun Microsystems, Inc.

NFS® is a registered trademark of Sun Microsystems, Inc.

UNIX® and OPEN LOOK® are registered trademarks of UNIX System Laboratories, Inc.

The X-Window System™ is a trademark of Massachusetts Institute of Technology.

OSF/Motif™ is a trademark of the Open Software Foundation, Inc.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations. SPARC International, Inc. disclaims any responsibility for specifying which trademarks are owned by which companies or organizations.

SPARC COMPLIANCE DEFINITION 2.3

SCD
2.3

Preface

Audience and Purpose	i
Organization and Content	i
Publication Conventions	i
Other Publication Conventions	i

CHAPTER 1: Introduction

Overview	1-1
Introduction Changes	1-1
Definitions of Terms	1-1
Interface Set	1-1
Interface Member	1-1
Interface	1-1
Required	1-1
Optional	1-1
Deprecated	1-2
Rationale	1-2
Experimental	1-2
Normative References	1-2
Relationship to other Standards	1-4
Future Direction	1-5
Upward Compatibility	1-5
Summary of Changes	1-5
Changes between SCD 2.2 and SCD 2.3	1-5
Structure of the SPARC Compliance Definition	1-7
System Feature Interfaces	1-7
Library Interfaces	1-7
Command Interface	1-8
Definition of SPARC Compliance	1-8
Conforming Implementations	1-8
Conforming Application Programs	1-8
Compliance Testing	1-9

CHAPTER 2: Software Installation

Overview	2-1
Software Installation Changes	2-1
CD-ROM Medium	2-1

CHAPTER 3: Low-Level System Information

Low-level System Information Changes	3-1
--------------------------------------------	-----

CHAPTER 4: Object Files

Object Files Changes	4-1
Relocation Types	4-1

CHAPTER 5: Program Loading and Dynamic Linking

Program Loading and Dynamic Linking Changes	5-1
---------------------------------------------------	-----

CHAPTER 6: Libraries

Overview	6-1
System Library Changes	6-2
System Library Changes (continued)	6-3
C Library Changes	6-4
Network Services Changes	6-5
System Data Interface Changes	6-6
Miscellaneous ABI Changes	6-7
Miscellaneous ABI Changes (continued)	6-8
The System Library	6-9
The libsys Interfaces	6-9
ABI Extensions	6-9
Long Long Intrinsics support	6-12
The C Library	6-13
The libc Interfaces	6-13
ABI Extensions	6-13
The Network Services Library	6-15
Overview	6-15
The libnsl Interfaces	6-15
ABI Extension	6-15
Networking Interface Set	6-17
Structures and Manifest Constants	6-17
Dynamic Object File Loading	6-23
Overview	6-23
ABI Extensions	6-23
Multithreading Library	6-24
Overview	6-24
Additional Interfaces in libsys/libc	6-24
ABI Extensions	6-24
Asynchronous I/O	6-27
Overview	6-27
ABI Extensions	6-27
The Math Library	6-28
Overview	6-28
The Large files Library	6-29
Overview	6-29

CHAPTER 7: Formats and Protocols

Formats and Protocols Changes	7-1
Interconnecting SCD Conforming Systems	7-1
Overview	7-1
Transport Providers	7-1
Additional Interfaces	7-1

CHAPTER 8: System Commands

Overview	8-1
System Commands Changes	8-1
System Commands Changes (continued)	8-2

CHAPTER 9: Execution Environment

Execution Environment Changes	9-1
-------------------------------------	-----

CHAPTER 10: Windowing and Terminal Interfaces

Windowing and Terminal Interfaces Changes	10-1
The X Library	10-1
Unsafe Macros	10-6
X Library Changes	10-9
X Library Changes (continued)	10-10
X Library Changes (continued)	10-11
The X Extension Library	10-20
Overview	10-20
The Extension Library Interfaces	10-20
The X Toolkit	10-21
Overview	10-21
The libXt Interfaces	10-21
Deprecated X Toolkit Functions	10-24
Subclassing Xt Widgets	10-37
The OPEN LOOK Widget Set	10-48
Overview	10-48
The libXol Interfaces	10-48
Motif 1.2 Widget Set	10-57
Overview	10-57
The Motif Interfaces	10-57

CHAPTER 11: Development Environments

Overview	11-1
Development Environments Changes	11-1

CHAPTER 12: Networking

Networking Changes	12-1
--------------------------	------

Index

SCD

2.3

Preface

Audience and Purpose

The SPARC International *SPARC Compliance Definition* (SCD) is intended for use by anyone who is creating binary compatible SPARC systems or applications.

The intended audience of the SCD documents consists of two groups: system and application developers. For system developers, the SCD provides a reference to those interfaces and features which must be supplied by a SPARC compliant system. For application developers, the SCD provides a reference to interfaces and features that may be relied upon in all SPARC compliant systems.

This publication is intended to fulfill the following purposes:

- *Identify areas beyond the System V Application Binary Interface (gABI) and the System V Application Binary Interface, SPARC Processor Supplement (psABI) that the SPARC community deems important.*
- *Address ambiguous and/or loose specifications in current ABI documents.*

Organization and Content

The SCD 2.3 has been divided into chapters, as follows:

Chapter 1	Introduction
Chapter 2	Software Installation
Chapter 3	Low-Level System Information
Chapter 4	Object Files
Chapter 5	Program Loading and Dynamic Linking
Chapter 6	Libraries
Chapter 7	Formats and Protocols
Chapter 8	System Commands
Chapter 9	Execution Environment
Chapter 10	Windowing and Terminal Interfaces
Chapter 11	Development Environments
Chapter 12	Networking
	Index

This new organization follows the organization of the *System V Application Binary Interface* and *System V Application Binary Interface, SPARC Processor Supplement* documents. Having a parallel organization makes this document easier to use than previous editions of the SCD.

Publication Conventions

This publication uses page format and typographic variances to highlight particular kinds of information. These conventions of usage are generally consistent with publication conventions used by other UNIX publications, such as the AT&T *System V Interface Definition*, Third Edition.

Other Publication Conventions

The following typographical conventions are used within the text of this publication:

- *Filenames, pathnames, and system messages are shown in:*
`typewriter font like this.`

- Titles of chapters in this publication are shown in plain Roman font, inside quotation marks like this:
"Introduction."
- Document titles are shown in plain, nonbold italic font like this:
System V Interface Definition (Third Edition).

CHAPTER 1: Introduction

SCD
2.3

Introduction

Overview

This document is version 2.3 of the SPARC Compliance Definition.

The SPARC Compliance Definition, or SCD, defines a set of interfaces that all SPARC Compliant systems must provide in their implementations. The SCD provides information for binary-level compatibility, encompassing both the *System V Application Binary Interface* (gABI), and the *System V Application Binary Interface*, SPARC Processor Supplement (psABI) documents.

Introduction Changes

The following are changes to the *System V application Binary Interface* as reported to SPARC International.

#	Facility	Location	Description
1	How to use the System V ABI	gABI	The math routines are also available as a shared resources

Definitions of Terms

Interface Set

The term “interface set” refers to a named collection of facilities, defined in the SPARC Compliance Definition, that is provided by a platform and can be used by an application. These collections, or “interface sets”, are listed in the section below titled “Structure of the SPARC Compliance Definition”.

An example is: the X11 Library Interface Set.

Interface Member

The term “interface member” is also used as a generic reference to any single facility that is provided by a platform for use by an application program.

Examples are: the printf function; the errno global data item.

Interface

The unadorned term “interface” means either “interface set” or “interface member” depending on the immediate context of its use. Any REQUIRED or OPTIONAL interface defined in this document will be part of the SCD for at least three years.

Required

The term “required” in this document is a qualifier for the terms “interface set,” “interface member,” and “interface.” When the term “REQUIRED interface set” is used in this document, SPARC conforming systems must provide the interface set; conformant applications can rely on the designated interface set always being available on any conforming system. The terms “REQUIRED interface member” and “REQUIRED interface” are defined similarly.

Optional

The term “optional” in this document is a qualifier for the terms “interface set,” “interface member,” and “interface.” When the term “OPTIONAL interface set” is used in this document, SPARC conforming systems may, but need not, supply the interface set; if a conforming system does supply the interface, the interface set must be present in its entirety, as defined by this document; applications can not rely on the designated interface set being available on any conforming

systems, but if the interface set is available on a particular conforming system, a conforming application can rely on the interface set being available in its entirety on that particular conforming system. The terms “OPTIONAL interface member” and “OPTIONAL interface” are similarly defined.

Deprecated

The term “deprecated” in this document is a qualifier for the terms “interface set,” “interface member,” and “interface.” When the term “DEPRECATED interface set” is used in this document, programmers are discouraged from using the designated interface set in new applications because the “DEPRECATED interface set” may not be supported in future versions of the SCD. The qualifier “deprecated” is orthogonal to the qualifiers “required” and “optional”. When an “interface set” is designated as “deprecated” the date of deprecation will be stated by the specification. “Interface sets,” marked as “deprecated,” will be kept in the SCD for at least three (3) years from the original deprecation date. The “DEPRECATED interface set” will also include in its specification, the earliest date at which the designated “interface set” may be removed from the specification. No required or optional interface will be removed from the standard without first being deprecated. The terms “DEPRECATED interface member” and “DEPRECATED interface” are defined similarly.

Rationale

Paragraphs labeled “rationale” in this document are non-normative and are for information only. An example of a Rationale paragraph follows below.

Rationale

The SPARC International Compliance and Compatibility Committee agreed that it would be more useful to intersperse rationale comments throughout the document than to confine them to an appendix.

Experimental

The term “experimental” in this document is a qualifier for the terms “interface set,” “interface member,” and “interface.” When the term “EXPERIMENTAL interface set” is used in the document, applications programmers are warned that 1) the designated interface set may not be available on any SPARC conforming systems, and, 2) the specification of the designated interface may change at any time or be deleted from the SCD at the sole discretion of SPARC International; SPARC International makes no commitment of a three-year stable period for any “EXPERIMENTAL interface set.”

Rationale

As an example, this release of the SPARC compliance definition includes the EXPERIMENTAL Large File Support Library. This interface set is completely new. Because the Large File Support Library is new, we have no experience with the correctness of the interface. Field experience may require that certain portions of the interface change to make the interface more useful or practical.

Normative References

The normative references called out in the SPARC Compliance Definition are:

- ***SCD 2.3 Interface Semantics***
SPARC International
- ***System V Application Binary Interface, Third Edition***
Unix Press (Prentice Hall), ISBN 0-13-100439-5
- ***System V Application Binary Interface SPARC Processor Supplement, Third Edition***
Unix Press (Prentice Hall), ISBN 0-13-104696-9
- ***The SPARC Architecture Manual, Version 8***
Prentice Hall, ISBN 0-13-825001-4
- ***System V Interface Definition, Third Edition, Volumes 1 - 5***
USL/Novell Select Code 320-136 (Volume 1), 320-137 (Volume 2), 320-138 (Volume 3),

320-139 (Volume 4),

Volume 1	Addison-Wesley	ISBN 0-201-56652-0
Volume 2	Addison-Wesley	ISBN 0-201-56653-0
Volume 3	Addison-Wesley	ISBN 0-201-56654-0
Volume 4	Addison-Wesley	ISBN 0-201-56655-0
Volume 5	Addison-Wesley	ISBN 0-201-56656-7

- ***The X Window System (Third Edition)***

by Robert W. Scheifler and James Gettys

Digital Press, ISBN 1-55558-088-2

- ***X Toolkit Intrinsics - C Language Interface***

by Joel McCormack, Paul Asente, and Ralph R. Swick

Distributed by the X consortium with the X Version 11, Release 5

available through FTP from export.lcs.mit.edu

- ***X11 Non-rectangular Window Shape Extension***

by Keith Packard

Copyright X Consortium

- ***OSF/Motif Programmer's Guide (Rel. 1.2- Revised)***

Prentice-Hall, ISBN 0-13-643115-1

Rational:

If the old reference is not available, a new Motif Reference can be used:

Motif Reference, X/Open CAE Specification:

Motif Toolkit API, ISBN 1-85912-024-5

Currently this Motif reference is different than the old one in the following:

- 1) Two functions (XmCreateCommandDialog, XmDropSiteRegistered) are added
- 2) Dozens of functions (e.g., MrmOpenHierarchy, XmStringCreateLtoR) are dropped.
- 3) Header files are not quite the same.

- ***OLIT Reference Manual***

Sun Microsystems, Part No. 800-6055-10, Revision A

- ***ISO 9660-1988: Volume and file structure of CD-ROM for information interchange***

1988-09-01

- ***ISO/IEC 10149: Data Interchange for read-only 120mm optical data disk (CD-ROM)***

1989-09-01

- ***RFC 1700***

URL: <http://info.internet.isi.edu/1s/in-notes/rfc/files>

The definition of each interface in the SPARC Compliance Definition may reference one or more of the above documents. In those cases, the portion of the normative reference that is called out is part of this standard.

The definition of each Interface in the SPARC Compliance Definition may list errata to any of the above documents. In each such listed erratum, the definition contained in the erratum supersedes the corresponding portion of the normative reference.

These documents may be acquired from most technical book stores; additionally, SPARC International provides

assistance in acquiring these references. If you require assistance in acquiring these references, call SPARC International at:

(415) 321-8692.

ISO documents can be ordered from:

International Organization for Standardization,
1 Rue de Varembe, Case Postale 56, CH-1211 Geneva 20 Switzerland,
(Tel) +41 22 34 12 40
URL: <http://www.iso.ch>

or ANSI (ISO member for the US):

ANSI, 11 W 42nd St. 13th floor, New York, NY 10036,
(Tel) 1-212-642-4900.

Prentice-Hall documents can be obtained at:

PTR Prentice Hall, Corporate Sales Department
113 Sylvan Avenue, Englewood Cliffs, New Jersey 07632
(Tel) (201) 592-2863 (bulk copies), (Tel)(515) 284-6761 (single copies).
(Fax) (201) 592-2249

Rationale

The SCD represents a proper super-set of the required interfaces and features described in the two ABI documents. One of the purposes of this document is to serve as the conduit through which features may migrate first into the processor specific ABI (SPARC psABI), and finally into the generic ABI (gABI). Consequently, the SCD includes a set of features and their associated interfaces that are beyond the ABI definitions. These features, and their associated interfaces have been included, in some cases to correct deficiencies in the ABI specifications, and in others to standardize functionality already in common use throughout the SPARC community.

Relationship to other Standards

As the SCD is a specification for binary level compatibility, it is important that it not conflict with already existing standards work, either de facto or de jure. To this end, the SCD 2.x draws upon the *System V Interface Definition* (Third Edition), (indirectly through references to the System V ABI) as the specification to which it will remain functionally consistent. As a consequence, the conformance of SCD 2.x to other standards documents/agencies is minimally the same as that of the *System V Interface Definition*. Examples of standards to which this pertains are:

- **POSIX 1003.1-1990** (ISO 9945-1) 1990 (E) (ISO/IEC) (IEEE/ANSI Std 1003.1-1990):

Information Technology - portable operating system interface (POSIX)

Part 1: System Application Program Interface (API)[C Language]

ISBN: 1-55937-061-7

- **X/Open Portability Guide, Issue 3 (XPG3)**

X/Open Portability Guide 1988 X/Open Company Limited

Vol1: XSI Commands and Utilities ISBN:0-13-685835-X □

Vol2: XSI System Interface and Headers ISBN:0-13-685843-0 □

Vol3:	XSI Supplementary Definitions	ISBN:0-13-685850-3
Vol4:	Programming Languages	ISBN:0-13-685868-6
Vol5:	Data Management	ISBN:0-13-685876-7
Vol6:	Window Management	ISBN:0-13-685884-8
Vol7:	Networking Services	ISBN:0-13-685892-9
set of 7 volumes		ISBN:0-13-685819-8

□ Referenced indirectly in the SVID.

Future Direction

1- Future direction for SCD is POSIX

2- non-POSIX is for existing implementations and, where they differ from posix, they are designated as EXPERIMENTAL

Upward Compatibility

The interfaces in SCD 2.3 are upwardly compatible with the interfaces in SCD 2.2, which in turn are upwardly compatible with the interfaces in SCD 2.1 and SCD 2.0. That is to say, an application written to the interfaces defined in SCD 2.0 will run successfully without change or re-compilation on a system that implements SCD 2.1, SCD 2.2, or SCD 2.3.

Summary of Changes

Beginning with this issue, the SPARC Compliance Definition has been reorganized to parallel the *System V Application Binary Interface Edition 3* and *System V Application Binary Interface, SPARC Processor Supplement Edition 3* documents from USL (Novell).

Changes between SCD 2.2 and SCD 2.3

General Changes:

- 1- All specifications in addition to new specification have been moved out of the SCD 2.2 and included into a new document called SCD 2.3 Interface semantics.
- 2- All references to the psABI and gABI revised Edition (2nd), have been updated to the 3rd Edition of these documents.
- 3- All errata have been modified and moved to the first page of their respective chapter.

Changes by Chapter:

Chapter 1:

- 1- The errata section has been moved to the beginning of the chapter
- 2- "Normative References" and "Relationship to other Standards" sections have been updated
- 3- "Additional Library Interfaces" section have been merged into this new section "Changes between SCD 2.2 and SCD 2.3."

Chapter 2:

- 1- Table 2-1 has been added which list the software installation commands required by the SCD 2.3

Chapter 3:

- 1- "Self Modifying coding Practice" section has been added to this chapter.
- 2- New errata section (Low-Level System Information Changes" has been added to this chapter.

Chapter 4:

- 1- "Relocation Types" section was added to this chapter.

Chapter 5:

- 1- "Shared Object Dependencies" errata was added to the "Program Loading and Dynamic Linking changes"

Chapter 6:

- 1- New functions and exported data have been added to libc, libsys, and libnsl section: These new functions/data are indicated in their respective tables and figures
- 2- Long long intrinsics section have been added to the libsys section
- 3- Three new libraries (libthread, libaio, and libm) were added.
- 4- "The Socket Library" section has been renamed "Networking Interface Set"

Chapter 7:

- 1- Reference changed from RFC 1340 to RFC 1700, in addition to minor corrections.

Chapter 8:

- 1-Nine new commands have been added to table 8-1.

Chapter 9:

- 1- The "Execution Environment Changes" section was updated

Chapter 10:

- 1- The chapter was fully re-organized
- 2- new "X Library Changes" errata sections has been added
- 3- Eight new functions has been added to libX content (Table 10-1).
- 4- "The LibXol Interfaces" section was updated and a new rationale was added.
- 5- "The Motif Interfaces" section was updated and a new rationale was added.
- 6- "The Windowing and Terminal Interfaces Changes" section was deleted.

Chapter 11:

- 1- This is a new chapter for the Development Environments

Chapter 12:

- 1- This is a new chapter for Networking.

Structure of the SPARC Compliance Definition

The Application Binary Interface defined by the SCD consists of a set of System Feature Interfaces, a set of Library Interfaces, and a Command Interface.

Each such named Interface is designated as either Required, Optional, or Experimental.

System Feature Interfaces

The System Feature Interfaces are:

- Object File Format
- Program Loading and Linking
- Low-level System Information
- Formats and Protocols
- Software Installation

Library Interfaces

Each Library Interface is a collection of facilities that is implemented as one or more shared objects. (Shared objects are defined in the Object File Format.)

The Library Interfaces are:

- System Library Interface
- C Library Interface
- Network Services Interface
- Socket Services Interfaces
- Network Address Resolution Library Interface
- Threads Interface
- Async I/O Interface
- Math Library
- Dynamic Linking Library Interface
- X Library Interface
- X Extensions Interface
- X Toolkit Library Interface
- Open Look Library Interface
- Motif 1.2 Library Interface
- Large File Support

Each Library Interface consists of

- Function entry points and their names
- Function arguments for each function entry point
- Global data and their names
- Manifest constants used in definitions of function arguments and global data
- Visible data structures used in function arguments and global data

- One or more shared objects, each having a particular name, each accessible through a particular pathname, and each containing the function entry points, function entry point names, global data, and global data names defined for that Library Interface.

Command Interface

The Command Interface is the set of commands available to application programs. The Command Interface is defined in the chapter titled “Commands”.

Definition of SPARC Compliance

The terms “SPARC-compliant” and “conforming” are used interchangeably in this document. Their meaning is:

Conforming Implementations

A conforming implementation is one that provides all of the Required Interfaces, in their entirety.

A conforming implementation may provide one or more of the Optional Interfaces. Each Optional Interface that is provided must be provided in its entirety. The product documentation must state which Optional Interfaces are provided.

A conforming implementation, when provided with standard data formats and values at a named interface, will provide the behavior defined for those values and data formats at that interface. However, a conforming implementation may consist of separately packaged and/or sold components. For example, a vendor of a conforming implementation might sell the hardware, operating system and windowing system as separately packaged items.

A conforming implementation may provide additional interfaces with different names. It may also provide additional behavior corresponding to data values outside the standard ranges, for standard named interfaces. Such additional interfaces, or additional inputs to standard interfaces, are called extensions to the standard. If an implementation provides extensions to the standard, its documentation must clearly identify the extensions as such.

Conforming Application Programs

A conforming application program has the following characteristics:

Its executable files are either Bourne shell scripts or object files in the format defined for the Object File Format System Interface.

Its object files participate in dynamic linking as defined in the Program Loading and Linking System Interface.

It employs only the instructions, traps, and other low-level facilities defined in the Low-Level System Interface as being for use by application programs.

It does not require or use any interface, facility, or implementation-provided extension that is not defined in this standard in order to be installed or to execute successfully.

If it requires any Optional Interface defined in this standard in order to be installed or to execute successfully, the requirement for that Optional Interface is stated in the application’s documentation.

It does not use any interface or data format that is not required to be provided by a conforming implementation; unless:

1. if any such interface or data format is used, it is generally available to anyone who wants to purchase or acquire it; and
2. if such an interface or data format is supplied by another program through direct invocation of that program during execution, that program is in turn a SPARC-compliant application; and
3. the use of that interface or data format, as well as its source, is identified in the documentation of the application program.

Rationale

A SPARC-compliant application is expected to have no dependencies on any vendor extensions to the standard. The most common such extensions are additional function entry points and additional libraries other than the ones defined in the SCD. If an application requires such extensions it is not portable, since other SCD-compliant platforms may not provide those extensions.

A SPARC-compliant application is required to use system services on the platform it's running on, rather than importing system routines from some other platform. Thus it must link dynamically to any routines in the platform that perform system traps to kernel services.

It is to be expected that some programs may be companion programs to other programs. For example, a query program may be a companion to a data base program; a pre-processor may be an adjunct to one or more compilers; a data re-formatter may convert data from one document manager to another. In such cases, the program may or may not be SPARC-compliant regardless of whether the other program it's dependent on is SPARC-compliant.

If such an application merely uses data produced by another program, the application's compliance is independent of the other program's compliance.

If such an application actually invokes another program during execution (as, for example, a third-party math library), the invoking program is SPARC-compliant only if it also constitutes a SPARC-compliant application in combination with the invoked program.

Compliance Testing

Test suites will be used in conjunction with this standard to verify the conformance of applications and platforms to this standard. Contact SPARC International for additional test suite information at (415)321-8692.

The System Compliance Test (SCT) will be used to verify a system's implementation of all the Interfaces defined in the SPARC Compliance Definition.

The SPARC Application Verifier (SAV) will be used to verify an application's adherence to the Interfaces defined in the SPARC Compliance Definition.

CHAPTER 2: Software Installation

SCD
2.3

Software Installation

Overview

Most information regarding software installation may be found in Chapter 2 of the gABI and Chapter 2 of the psABI. The commands supported are listed in table 2.1 below. This section is an addendum to Chapter 2, page 2-1, of the psABI. This section adds support for using CD-ROM medium for physical distribution of SCD-conforming software. It is an OPTIONAL INTERFACE. If software is distributed on CD-ROM, it must be in one of the formations specified below. SPARC-compliant systems need support for CD-ROM, some vendors are already shipping all their software on CD-ROM's.

Table 2-1. Software Installation commands

<code>installf</code>	<code>pkgask</code>	<code>pkginfo</code>	<code>pkgrm</code>
<code>pkgadd</code>	<code>pkgchk</code>	<code>pkgparam</code>	<code>removef</code>

Software Installation Changes

The following are changes to the *System V Application Binary Interface*, and the *System V Interface Definition (Third Edition)* as reported to SPARC International.

#	Facility	Location	Description
1	pkginfo(AS_CMD)	SVID, Vol. II	Delete "-r" from the list of supported options for pkginfo.
2	pkgadd(AS_CMD)	gABI	Change page 2-13 of the gABI to specify that the request script, if provided, runs with a uid of root and that standard input is attached to /dev/null.
3	The request script Procedure Scripts	gABI	The description about the execution environment is incorrect. Only uid == root is guaranteed.

CD-ROM Medium

CD-ROM medium recorded in the format specified in *ISO/IEC 10149: Data Interchange for read-only 120mm optical data disk (CD-ROM)* is added to the list of approved media on page 2-1 of the *System V Application Binary Interface, SPARC Processor Supplement*.

The information on the media must be represented either

- serially as the data stream created using `dd(AU_CMD)` or `cpio(BU_CMD)` utilities; or
- as file structured data that must be represented as described in *ISO 9660: 1988 - Volume and file structure of CD-ROM for information interchange*.

Rationale

The most common format for CD-ROM's is the ISO 9660 format, which supports MS-DOS filesystem semantics only. The ISO 9660 format is robust and stable, and has a huge installed base. That is why the ISO 9660 format has been included in SCD2.3 as an OPTIONAL standard for SPARC-compliant systems.

Support for ISO 9660 format CD-ROM's is already available from several other operating system vendors.

Since the restrictions placed on a filesystem by the ISO 9660 format are too restrictive for most UNIX users, a POSIX conforming filesystem is needed. The Rock Ridge Interchange Protocol was created to fill this gap. The Rockridge filesystem is actually an extension to (and compliant with) the ISO 9660 specification.

The Rock Ridge filesystem appears to be stable at this time. However there are some issues concerning bootability, security, and sparse files which are still being addressed by the IEEE working group on CD-ROM filesystems. There will be some minor changes made before the Rock Ridge filesystem is adopted as a NIST (National Institute of Science and Technology) standard.

For these reasons, the Rock Ridge filesystem is being excluded from SCD 2.3.

Upon adoption as a standard by NIST, it is expected that the Rock Ridge format will be included in the standard.

CHAPTER 3: Low-Level System Information

SCD
2.3

Low-Level System Information

Low-level system information pertinent to SPARC platforms may be found in Chapter 3 of the *System V ABI, SPARC Processor Supplement*. Information such as page size restrictions, as well as stack management, function calling sequence and data representations may be found there.

Self-Modifying Coding Practices:

Self modified (or otherwise changed) code sequence must be the target of the appropriate sequence of FLUSH instructions prior to being executed. A specific example of a problematic example can be seen in the code fragment:

```
(void) read(fd, buf, sizeof (buf));
(*(void(*)())buf)();
```

which treats the contents of “buf” as a function which has just been read in.

Low-level System Information Changes

#	Facility	Location	Description																				
1	Fundamental Types	psABI	<div>Add the following to Figure 3-1:</div> <table><tr><td>Type</td><td>C</td><td>sizeof</td><td>Alig.</td><td>SPARC</td></tr><tr><td>Integral long long</td><td></td><td>8</td><td>8</td><td>signed doubleword</td></tr><tr><td>signed long long</td><td></td><td>8</td><td>8</td><td></td></tr><tr><td>unsigned long long</td><td></td><td>8</td><td>8</td><td>unsigned doubleword</td></tr></table>	Type	C	sizeof	Alig.	SPARC	Integral long long		8	8	signed doubleword	signed long long		8	8		unsigned long long		8	8	unsigned doubleword
Type	C	sizeof	Alig.	SPARC																			
Integral long long		8	8	signed doubleword																			
signed long long		8	8																				
unsigned long long		8	8	unsigned doubleword																			
2	Registers and the stack Frame	psABI	<div>Add the following description to the Stack Frame page 3-13:</div> <div>%i0,%i1,%o0 and %o1 64-bit integer return values appear and%o1 in%i0 and%i1 (most significant word in%i0). A calling function receives values in the coincident out registers,%o0 and%o1.</div>																				
3	Integral and Pointer Arguments	psABI	<div>Add the following description:</div> <div>64-bit integer argument uses two registers.</div>																				
4	Functions Returning Scalars or No Value	psABI	<div>Add the following description:</div> <div>A function that returns a 64-bit integer value places its result in%i0 and%i1 (most significant word in%i0); the calling function finds that value in%o0 and%o1.</div>																				

CHAPTER 4: Object Files

SCD

2.3

Object Files

Processor independent descriptions of the object file format for System V Release 4 may be found in Chapter 4 of the *System V ABI*. Information specific to SPARC platforms may be found in Chapter 4 of the *System V ABI, SPARC Processor Supplement*.

Object Files Changes

The following are changes to the *System V Application Binary Interface* as reported to SPARC International.

#	Facility	Location	Description
1	SHT_DYNSYM	gABI	On page 4-14 add before the last sentence: "However this minimal set of symbols will always include all symbols of STB_GLOBAL binding."

Relocation Types

The following table augments the relocation types found in Chapter 4 of the *System V ABI, SPARC Processor Supplement*. The format and interpretation of these entries are as described in that document. These relocation types are, in their entirety, introduced as an EXPERIMENTAL INTERFACE.

Figure 4-1. Relocation Types

Name	Value	Field	Calculation
R_SPARC_PLT32	24	V-word32	L + A
R_SPARC_HIPLT22	25	T-imm22	(L + A) >> 10
R_SPARC_LOPLT10	26	T-simm13	(L + A) & 0x3ff
R_SPARC_PCPLT32	27	V-word32	L + A - P
R_SPARC_PCPLT22	28	V-disp22	(L + A - P) >> 10
R_SPARC_PCPLT10	29	V-simm12	(L + A - P) & 0x3ff
R_SPARC_10	30	V-simm10	S + A
R_SPARC_11	31	V-simm11	S + A
R_SPARC_64	32	V-xword64	S + A
R_SPARC_OLO10	33	V-simm13	((S + A) & 0x3ff) + O
R_SPARC_HH22	34	V-imm22	(S + A) >> 42
R_SPARC_HM10	35	T-simm13	((S + A) >> 32) & 0x3ff
R_SPARC_LM22	36	T-imm22	(S + A) >> 10
R_SPARC_PC_HH22	37	V-imm22	(S + A - P) >> 42
R_SPARC_PC_HM10	38	T-simm13	((S + A - P) >> 32) & 0x3ff
R_SPARC_PC_LM22	39	T-imm22	(S + A - P) >> 10
R_SPARC_WDISP16	40	V-d2/disp14	(S + A - P) >> 2
R_SPARC_WDISP19	41	V-disp19	(S + A - P) >> 2
R_SPARC_GLOB_JMP	42	V-xword64	S + A
R_SPARC_7	43	V-imm7	S + A
R_SPARC_5	44	V-imm5	S + A
R_SPARC6	45	V-imm6	S + A

The descriptive portions of this table are for background information only. The primary significance of the table, given the presence of these interfaces as EXPERIMENTAL is to reserve the space of relocation values indicated in the table for use in EXPERIMENTAL system implementations. No conforming application will employ these values for any purpose, and no system is required to demonstrate conformance to any interpretation of these relocation types.

CHAPTER 5: Program Loading and Dynamic Linking

SCD
2.3

Program Loading and Dynamic Linking

Processor independent descriptions of program loading and linking for SCD compliant systems may be found in Chapter 5 of the *System V Application Binary Interface*. Information specific to the SPARC platforms may be found in Chapter 5 of the *System V Application Binary Interface, SPARC Processor Supplement*.

Program Loading and Dynamic Linking Changes

The following are changes to the *System V Application Binary Interface*, and the *System V Application Binary Interface, SPARC Processor Supplement* as reported to SPARC International.

#	Facility	Location	Description
1	LD_LIBRARY_PATH	gABI	Change the order of the first and the second bullets in page 5-20 such that the influence of LD_LIBRARY_PATH takes precedence over DT_RPATH specifications.
2	Dynamic Linking	gABI	Add a new third bullet to the entries on page 5-20: "DT_RPATH specifications influence search operations for their own DT_NEEDED objects. Each evaluation of a given object's set of DT_NEEDED specification uses <i>that object's</i> DT_RPATH. Thus, if an executable specifies a set of DT_NEEDED objects (e.g., a , b , and c) and a DT_RPATH specification of x:y , then the search for a , b , and c will involve the paths x and y . If, when later evaluating the DT_NEEDED object for a (e.g., d), then x and y will not be used for that search unless a also specifies a DT_RPATH containing them."
3	Initialization and Termination Functions	gABI	Add the following new third paragraph on page 5-22: "Initialization and Termination functions can expect to use all libsys and libc ABI-defined services in their execution."
4	Shared Object Dependencies	gABI	In page 5-20, the gABI specifies in a "NOTE" that for setuser and set-group ID programs, LD_LIBRARY_PATH is ignored and DT_RPATH entries are used. This statement is incomplete. DT_RPATH entries should be used only to the extent that components beginning with "/" are acceptable (not relative path), in particular, relative path names are not used as these constitute a security hazard. Further, the prohibition against LD_LIBRARY_PATH is unnecessarily restrictive and conflicts with at least some widespread existing practice, in which those items contained in LD_LIBRARY_PATH which are also acceptable DT_RPATH entries or are "/usr/lib" are also used.
5	Dynamic linking	psABI	Add a section entitled "Dynamic Linker" as the first subsection of the "Dynamic Linking" section, which is: "The value of the program header element PT_INTERP in an ABI-conforming program is the reference name for libsys. As a special case, the reference name for version 1 of the C library reference name is also accepted as a legitimate PT_INTERP specification."

SCD

2.3

Libraries

Overview

This chapter defines the interfaces set represented by each of the following libraries:

- System Library
- C Library
- Network Services Library
- Socket Library
- Dynamic Object File Loading Library
- Large File Library
- Multithreading Library
- Asynch I/O Library
- Math Library

The System Library, the C Library, the Network Services Library, the Socket Library, the Dynamic Object File Loading Library, the Multithreading Library, the Asynchronous File I/O Library, and the Math Library are REQUIRED interface sets. The Large File Library is an EXPERIMENTAL interface set and conforming systems need not supply it.

Some of the entries in the tables which define the function interfaces provided by various libraries have a superscript. All entries with a superscript have an entry in the changes table describing differences between the SCD definition and the gABI, psABI, or *System V Interface Definition, Third Edition* definition of the function.

The first part of this chapter is the changes to the *System V Application Binary Interface, SPARC Processor Supplement*, and the *System V Interface Definition* as reported to SPARC International.

System Library Changes

#	Facility	Location	Description
1	__dtou	psABI	Change - On page 6-6, replace the description of exceptions for __dtou with "If $-2^{31} \leq a < 2^{32}$ then the operation is successful. If a is not a whole number, the inexact exception is raised. Otherwise, the value returned by __dtou is unspecified, and the invalid exception is raised. Note that negative values of a , in a successful operation, are first converted to integer and then cast to an unsigned integer."
2	__ftou	psABI	Change - On page 6-7, replace the description of exceptions for __ftou with "If $-2^{31} \leq a < 2^{32}$ then the operation is successful. If a is not a whole number, the inexact exception is raised. Otherwise, the value returned by __ftou is unspecified, and the invalid exception is raised. Note that negative values of a , in a successful operation, are first converted to integer and then cast to an unsigned integer."
3	_Q_qtou	psABI	Change - On page 6-5, replace the description of exceptions for _Q_qtou with "If $-2^{31} \leq a < 2^{32}$ then the operation is successful. If a is not a whole number, the inexact exception is raised. Otherwise, the value returned by _Q_qtou is unspecified, and the invalid exception is raised. Note that negative values of a , in a successful operation, are first converted to integer and then cast to an unsigned integer."
4	_environ	gABI	Addition - On page 6.6, add the symbol _environ to Figure 6-5.
5	Additional Entry Points	psABI	Page 6-5 of the <i>System V Application Binary Interface</i> states "ABI-conforming systems must provide a libsys entry point for each of [fstat, lstat, mknod, stat, and uname]. The name and syntax of [these entry points] may be the same as those characteristics of the source-level service or they may vary across processor architectures. The actual names of the entry points are specified in each processor's supplement to the ABI, together with the entry points' syntax information if names differ from those of the source-level services." The <i>System V Application Binary Interface, SPARC Processor Supplement</i> (psABI) is missing the required specification. A section titled <i>Additional Entry Points (Processor -Specific)</i> should be added to the beginning of chapter 6 of the psABI which states "The binary entry points for fstat, lstat, mknod, stat, uname exist with these names and with the same calling sequence as described in their source-level interface. Synonyms exist for each of these entry points."
6	errno	gABI	Addition - On page 6-6, add the symbol errno to Figure 6-5.
7	fcntl(BA_OS)	SVID, Vol. 1	Add a description of the command F_FREESP which reads: "Free storage space associated with a section of the ordinary file <i>fil</i> des. The section is specified by a variable of data type struct flock pointed to by the third argument <i>arg</i> . <i>l_whence</i> is SEEK_SET, SEEK_CUR, or SEEK_END to indicate that the relative offset <i>l_start</i> will be measured from the start of the file, the current position, or the end of the file, respectively. <i>l_start</i> is the offset from the position specified in <i>l_whence</i> . <i>l_len</i> is the size of the section. An <i>l_len</i> of 0 frees up to the end of the file; in this case, the end of file (i.e., file size) is set to the beginning of the section freed. Any data previously written into this section is no longer accessible."
8	fcntl(BA_OS)	SVID, Vol. 1	Change - The EAGAIN error return value only applies to files for which mandatory locking is enabled.

System Library Changes (continued)

#	Facility	Location	Description
9	getcwd(BA_OS)	SVID, Vol.1	Change - For POSIX conformance, the type of the second argument size should be size_t rather than int.
10	getgrent(BA_LIB)	gABI	Add the functions getgrent, setgrent, endgrent and fgetgrent to Figure 6-2 on page 6-4.
11	getgrent(BA_LIB)	SVID, Vol. 1	The description that the information in the group structure comes from the /etc/group file is too restrictive; the information may come from other sources. These sources are collectively called "group database". Applications should not depend on the implementation of the group database.
12	getpwent(BA_LIB)	gABI	Add the functions getpwent, setpwent, endpwent and fgetpwent to Figure 6-2 on page 6-4.
13	getpwent(BA_LIB)	SVID, Vol. 1	The description that the information in the passwd structure comes from /etc/passwd file is too restrictive; the information may come from other sources. These sources are collectively called "user database". Applications should not depend on the implementation of the user database.
14	Global Data Symbols	gABI	Change the description of _altzone. Replace "tzset(BA_LIB)" with "tzset(). See ctime(BA_LIB)."
15	mmap(KE_OS)	SVID, Vol. 1	Add to the paragraph which begins "Not all implementations..." insert "No implementation will permit an access to succeed where PROT_NONE has been set." after "... where PROT_WRITE has not been set."
16	read, readv(BA_OS)	SVID, Vol. 1	Addition - The SVID specifies that the write, writev(BA_OS)length of the struct iov[] in calls to readv()/writev() must be in the range 0 ≤ iovcnt ≤ IOV_MAX. However, IOV_MAX is never defined. SCD compliant systems will support a minimum of 16 elements in a struct iov[].
17	rename	gABI	Change - On page 6-4, move rename from Figure 6-2 to Figure 6-3.
18	sbrk	SVID, Vol. 1	Add description of the function sbrk. See the man page for this function in the SCD 2.3 Interface Semantics.
19	sbrk	gABI	Add the function sbrk to Figure 6-2 on page 6-4.
20	symlink(BA_OS)	SVID, Vol. 1	Change description of ENAMETOOLONG to "if the length of path2 exceeds {PATH_MAX}, or pathname component of path2 is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect."
21	system(BA_OS)	SVID, Vol.1	Change - For POSIX conformance, system() will ignore SIGINT and SIGQUIT, and block SIGCHLD while waiting for the command it invokes to terminate. Receipt of these signals will not result in system() returning with a -1 result and with errno set to EINTR.
22	waitid(BA_OS)	SVID, Vol. 1	Change - The flag WTRACED should be replaced with WTRAPPED.

C Library Changes

#	Facility	Location	Description
1	crypt(BA_LIB)	gABI	Add the function crypt to Figure 6-7 on page 6-10.
2	crypt(BA_LIB)	gABI	Add the function encrypt to Figure 6-7 on page 6-10.
3	crypt(BA_LIB)	gABI	Add the function setkey to Figure 6-7 on page 6-10.
4	fdopen(BA_OS)	SVID, Vol. 1	Change - The requirement that the <i>filides</i> argument be open is incorrect.
5	getitimer(RT_OS)	SVID, Vol.3	Change - The description of canonical form is incorrect. The microsecond value can be zero. Hardware platforms must provide at least 60 Hz resolution. Platforms may provide greater than 60 Hz resolution, but applications that rely on a faster clock will not be portable.
6	getitimer(RT_OS)	gABI	Add the function setitimer to Figure 6-7 on page 6-10.
7	gettimeofday(RT_OS)	gABI	Add the function gettimeofday to Figure 6-7 on page 6-10.
8	lockf(BA_OS)	SVID, Vol. 1	Addition - The EAGAIN error return value only applies to files for which mandatory locking is enabled.
9	sysinfo	gABI	Add the function sysinfo to Figure 6-7 on page 6-1
10	termios(BA_OS)	SVID, Vol.1	Change - On page 6-152, in the description of tcsetattr(), "If duration is not zero, zero-valued bits are not transmitted" is incorrect. Zero valued bits will be sent for implementation dependent period of time.

Network Services Changes

#	Facility	Location	Description
1	netconfig(RS_ENV)	SVID, Vol. 3	Change - On page 17-20 the type declaration of <code>nc_flag</code> should be changed from <code>char *</code> to <code>unsigned long</code> .
2	rpc_broadcast_exp	gABI	Add the function <code>rpc_broadcast_exp</code> to Figure 6-11 on page 6-13.
3	rpc_clnt_calls(RS_LIB)	SVID, Vol. 3	Change - On page 18-11 the function prototype of <code>rpc_call()</code> should be <code>rpc_call (char *host, u_long prognum, u_long versnum, u_long procnum, xdrproc_t inproc, char *in, xdrproc_t outproc, char *out, char *nettype)</code>
4	rpc_svc_err(RS_LIB)	SVID, Vol. 3	<p>Change - Description of the function <code>svcerr_progvers()</code> is missing its last two arguments in the function prototype. Prototype should be:</p> <pre>void svcerr_progvers(const SVCXPRT *xp, ulong_t low, ulong_t high)</pre> <p>where <i>low</i> and <i>high</i> represent the lowest and highest, respectively, of the versions of the service provided.</p>
5	svc_fdset	gABI	<code>svc_fds</code> in Figure 6-12 should be changed to <code>svc_fdset</code> .
6	t_alloc(BA_LIB)	SVID, Vol. 1	Change the sentence starting with "If the size value associated with any specified field is -1 or -2..." to "If the size value associated with any specified field is -1, <code>t_alloc()</code> will allocate the buffer with the size of 1024 bytes. If the size value is -2, <code>t_alloc()</code> will set the buffer pointer to NULL and the buffer maximum size to 0 and will return with success."
7	t_getstate(BA_LIB)	SVID, Vol. 1	Delete the phrase beginning with "or <code>t_getstate()</code> was called..."

System Data Interface Changes

#	Facility	Location	Description
1	<dirent.h>	psABI	Change the declaration of DIR, in Figure 6-5, to be an opaque type. Application programs can know neither the size nor the layout of this type.: <pre>typedef struct{ /* unspecified */ } DIR;</pre>
2	<fcntl.h>	psABI	The following manifest constant is needed for implementing ftruncate() and truncate() operations but is missing from Figure 6-7: <pre>#define F_FREESP 11</pre>
3	<rpc.h>	psABI	page 6-31, The identity of this header file is incorrect, it is specified in the SVID (and in existing practice) to be <rpc/rpc.h>.
4	<rpc.h>	psABI	Change - Delete the definition of RPC_ANYSOCK, and change the definition of RPC_ANYFD to be -1
5	<signal.h>	psABI	Change - On page 6-41, in Figure 6-33 for the struct sigaction type declaration change sigdisp_t sa_disp to void (*sa_handler)(). <pre>struct sigaction { int sa_flags; void (*sa_handler)(); sigset_t sa_mask; int sa_resv[2]; };</pre>
6	<signal.h>	psABI	page 6-41, The values “28” (SIGVTALRM) and “29” (SIGPROF) are missing from <signal.h> in the psABI. These are needed now that getitimer() and setitimer() are part of the SCD 2.3 (figure 6-33).
7	<signal.h>	psABI	page 6-41, The signal of values 32 and above are reserved to the system implementation and must not be used by an SCD-compliant application.
8	<sys/param.h>	psABI	Remove definition of HZ from Figure 6-23.
9	<sys/tiuser.h>	psABI	Change - In Figure 6-52 through 6-58 on page 6-59 through 6-63, header name <sys/tiuser.h> should be changed to <tiuser.h>.
10	<sys/types.h>	psABI	Addition - On page 6-63, in Figure 6-59, add the following type definitions: <pre>typedef unsigned int u_int; typedef unsigned long u_long; typedef unsigned short u_short; typedef char *caddr_t;</pre>
11	<wait.h>	psABI	Change - In Figure 6-66 on page 6-68, header name <wait.h> should be changed to <sys/wait.h>.

Miscellaneous ABI Changes

#	Facility	Location	Description
1	Dependencies Among Libraries	gABI	<p>Change - On page 6-2, at the statement which begins “Application executable and shared object files...” replace to the end of the paragraph with “Application executables must provide a complete list of those shared objects which the application uses directly. Each system library must supply a complete dependency graph for its own execution as DT_NEEDED entries.</p> <p>Rationale: No application should be required to know what secondary dependencies any platform system library may have. Such dependencies may vary from system to system.</p>
2	Shared Library Names	psABI	<p>Addition - A section should be inserted that identifies the actual version numbers and reference names for shared objects on a SPARC system.</p>

Table 6-1 Library Logical and Reference Names

Library	Reference Name
libc	/usr/lib/libc.so.1
libdl	/usr/lib/libdl.so.1
libnsl	/usr/lib/libnsl.so.1
libsocket	/usr/lib/libsocket.so.1
libsys	/usr/lib/ld.so.1
libm	/usr/lib/libm.so.1
libthread	/usr/lib/libthread.so.1
libaio	/usr/lib/libaio.so.1
liblf	/usr/lib/liblf.so.1
libX	/usr/lib/libX11.so.5 /usr/lib/libX11.so.4 (deprecated)
libXext	/usr/lib/libXext.so.0
libXt	/usr/lib/libXt.so.5 /usr/lib/libXt.so.4 (deprecated)
libXol	/usr/lib/libXol.so.3
libXm	/usr/lib/libXm.so.1.2
libMrm	/usr/lib/libMrm.so.1.2

3	Shared Library Names	gABI	<p>Deletion - Delete Table 6-1 on page 6-2.</p> <p>Addition - Actual full path names (reference names) of these shared libraries are specified in the appropriate processor supplement.</p>
---	----------------------	------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Miscellaneous ABI Changes (continued)

#	Facility	Location	Description
4	Shared Library Names	gABI	<p>Addition page 6-2 - A second paragraph should be inserted to this section that states: "The version numbers of shared objects are set on a per-processor basis with the constraint that they are derived from a Generic ABI 'reference version number' for each interface and must change their current value whenever that reference version number changes. In this manner, the reference names can reflect the often combined generic and processor specific portions of the interface in a consistent manner." "A shared object version number must change whenever one or more of the following occurs:</p> <ul style="list-style-type: none">• an entry point is deleted,• an entry point is added,• an entry point is changed,• program visible semantic properties change, or• changes to exported data objects change in size, type, or name."

The System Library

The libsys Interfaces

This section contains the REQUIRED libsys interfaces to basic system services listed in the *System V Application Binary Interface* and described in sections BA_OS, BA_LIB, BA_ENV, KE_OS, and RT_OS of the *System V Interface Definition, Third Edition*.

Effective November 1st, 1993, the sbrk function interface is DEPRECATED. The interface may be removed from the SCD as early as November 1st, 1996.

The interfaces listed below in Table 6-2, Table 6-3, Table 6-4, and Table 6-5 have been included in SCD 2.3 because they are REQUIRED to be present in the system library libsys. libsys is the entity obtained through the use of the *reference name* /usr/lib/ld.so.1. Table 6-3 contains the exported data which are also REQUIRED to be present in /usr/lib/ld.so.1. The format of these entries is: data[size], where size is a hexadecimal byte count. Issues regarding synonyms and global data symbols associated with this library can be found in the *System V Application Binary Interface*.

ABI Extensions

The SCD requires /usr/lib/libld.so.1 to have functions which are not specified by the gABI. These extra functions are either not defined in the SVID, or, are defined differently in the SCD than in the SVID. The semantics pages for these additional/modified function definitions are available in the SCD 2.3 Interface Semantics document.

Table 6-2 libsys Contents

_exit	getpgrp	realloc	symlink ¹
access	getpwent ^{1,2}	remove	sync
acct	getpid	rename ¹	sysconf
alarm	getpmsg	rewinddir	system ¹
atexit	getppid	rmdir	telldir
calloc	getpwnam	sbrk ¹	time
catclose	getpwuid	seekdir	times
catgets	getrlimit	semctl	ttynname
catopen	getsid	semget	ulimit
chdir	gettxt	semop	umask
chmod	getuid	setcontext	umount
chown	grantpt	setgid	uname
chroot	initgroups	setgrent ^{1,2}	unlink
close	ioctl	setgroups	unlockpt
closedir	isastream	setlocale	utime
creat	kill	setpgid	wait
dup	lchown	setpgrp	waitid ¹
endgrent ^{1,2}	link	setpwent ^{1,2}	waitpid
endpwent ^{1,2}	localeconv	setrlimit	write ¹
execl	lseek	setsid	writerv ¹
execle	lstat	setuid	
execvp	malloc	shmat	1- see system
execv	makecontext ²	shmctl	library changes at
execve	memcntl	shmdt	the beginning of
execvp	mkdir	shmget	this chapter.
exit	mknod	sigaction	2- New additions:
fattach	mlock	sigaddset	not in SCD 2.2
fchdir	mmap ¹	sigaltstack	
fchmod	mount	sigdelset	
fchown	mprotect	sigemptyset	
fcntl ¹	msgctl	sigfillset	
fdetach	msgget	sighold	
fgetgrent ^{1,2}	msgrcv	sigignore	
fgetpwent ^{1,2}	msgsnd	sigismember	
fork	msync	siglongjmp	
fpathconf	munlock	signal	
free	munmap	sigpause	
fstat	nice	sigpending	
fstatvfs	open	sigprocmask	
fsync	opendir	sigrelse	
ftok	pathconf	sigsend	
getcontext	pause	sigsendset	
getcwd ¹	pipe	sigset	
getegid	poll	sigsetjmp	
geteuid	profil	sigsuspend	
getgid	ptrace	stat	
getgrent ^{1,2}	ptsname	statvfs	
getgrgid	putmsg	stime	
getgrnam	putpmsg	strcoll	
getgroups	read ¹	strerror	
getlogin	readdir	strftime	
getmsg	readlink	strxfrm	
getpgid	readv ¹	swapcontext ²	

Table 6-3 Exported data for libsys

<code>_ctype[0x209]</code>	<code>_environ[0x4]</code> ^{1,2,3}	<code>daylight[0x4]</code>	<code>tzname[0x8]</code>
<code>__huge_val[0x8]</code>	<code>_numeric[0x2]</code>	<code>environ[0x4]</code> ²	
<code>_altzone[0x4]</code>	<code>_timezone[0x4]</code>	<code>errno[0x4]</code> ^{1,3}	
<code>_daylight[0x4]</code>	<code>_tzname[0x8]</code>	<code>timezone[0x4]</code>	

1 - see system library changes at the beginning of this chapter.

2- Conformant systems are required to reserve space for `_environ` in libsys. It is the responsibility of either the compilation system or the application to ensure that `_environ` is properly initialized. (see system library changes at the beginning of this chapter)

3- New additions: not in SCD 2.2

Table 6-4 libsys SPARC Support Routines

<code>.div</code>	<code>_Q_add</code>	<code>_Q_fne</code>	<code>_Q_sub</code>
<code>.mul</code>	<code>_Q_cmp</code>	<code>_Q_itoq</code>	<code>_Q_utoq</code>
<code>.rem</code>	<code>_Q_cmpe</code>	<code>_Q_mul</code>	<code>__dtou¹</code>
<code>.stret1</code>	<code>_Q_div</code>	<code>_Q_neg</code>	<code>__ftou¹</code>
<code>.stret2</code>	<code>_Q_dtoq</code>	<code>_Q_qtod</code>	
<code>.stret4</code>	<code>_Q_feq</code>	<code>_Q_qtoi</code>	
<code>.stret8</code>	<code>_Q_fge</code>	<code>_Q_qtos</code>	
<code>.udiv</code>	<code>_Q_fgt</code>	<code>_Q_qtou¹</code>	
<code>.umul</code>	<code>_Q_fle</code>	<code>_Q_sqrt</code>	
<code>.urem</code>	<code>_Q_flt</code>	<code>_Q_stoq</code>	

1- see system library changes at the beginning of this chapter.

Long Long Intrinsics support

libsys contains the following routines for support of operation on a 64-bit integer ("long long") for both signed and unsigned quantities. Their descriptions are available in the SCD 2.3 Interface Semantics document. Calling sequence for 64-bit integer arguments and return value is described in Low-level System Information Changes in Chapter 3.

Table 6-5 libsys Long Long Intrinsics support¹

<code>__mul64</code>	<code>__rem64</code>	<code>_Q_qtoll</code>	<code>_Q_lltoq</code>
<code>__umul64</code>	<code>__urem64</code>	<code>__dtoull</code>	<code>_Q_ulltoq</code>
<code>__div64</code>	<code>__dtoll</code>	<code>__ftoull</code>	
<code>__udiv64</code>	<code>__ftoll</code>	<code>_Q_qtoull</code>	

1- New additions: not in SCD 2.2

The long long type is supported in `printf(BA_LIB)` and `scanf(BA_LIB)` as follows:

In format string, conversion specifiers `d`, `i`, `o`, `u`, `x`, and `X` may be preceded by `ll` (ell ell) to indicate that corresponding argument is of type long long integer (`printf`) or pointer to long long integer (`scanf`).

The C Library

The libc Interfaces

This section contains the REQUIRED libc interfaces listed in the *System V Application Binary Interface* and described in the *System V Interface Definition, Third Edition*. Interfaces listed below in Table 6-4 have been included because they are REQUIRED on all systems conforming to the *System V Application Binary Interface*, made available through the reference name `/usr/lib/libc.so.1`. Table 6-6 contains the exported data which are also REQUIRED to be present in `/usr/lib/libc.so.1`. The format of these entries is: `data[size]`, where `size` is hexadecimal byte count. Issues regarding synonyms and global data symbols associated with this library can be found in the *System V Application Binary Interface*.

Table 6-6 Exported Data for libc

<code>__iob[0x140]</code>	<code>getdate_err[0x4]</code>	<code>opterr[0x4]</code>	<code>optopt[0x4]</code>
<code>_getdate_err[0x4]</code>	<code>optarg[0x4]</code>	<code>optind[0x4]</code>	

ABI Extensions

The SCD requires `/usr/lib/libc.so.1` to have functions which are not specified by or different from the gABI. These extra functions are either not defined in the SVID, or, are defined differently in the SCD than in the SVID. These functions are `crypt`, `setkey`, `encrypt`, and `sysinfo`. The semantics for these additional/modified function definitions are available in the SCD 2.3 Interface Semantics document.

Figure 6-1 contains manifest constants for `sysinfo`.

Rationale

The library version number has remained 1, as these functions are correctly included in existing SCD conformant systems.

Figure 6-1 Manifest Constants from `<sys/systeminfo.h>`

```
/* Commands to sysinfo() */

#define SI_SYSNAME      1      /* return name of operating system */
#define SI_HOSTNAME     2      /* return name of node */
#define SI_RELEASE      3      /* return release of operating system */
#define SI_VERSION      4      /* return version field of utsname */
#define SI_MACHINE      5      /* return kind of machine */
#define SI_ARCHITECTURE 6      /* return instruction set arch */
#define SI_HW_SERIAL    7      /* return hardware serial number */
#define SI_HW_PROVIDER  8      /* return hardware manufacturer */
#define SI_SRPC_DOMAIN  9      /* return secure RPC domain */
```

Table 6-7 Libc contents

__assert	getchar	nextafter ²	tcgetpgrp
__filbuf	getdate	nftw	tcgetsid
__flsbuf	getenv	nl_langinfo	tcsendbreak
_cleanup ²	getopt	pclose	tcsetattr
_tolower	getpass	perror	tcsetpgrp
_toupper	gets	popen	tdelete
_xftw	getsubopt	printf	tell
abort	getitimer ¹	putc	tempnam
abs	gettimeofday ¹	putchar	tfind
addseverity	getw	putenv	tmpfile
asctime	gmtime	puts	tmpnam
atof	hcreate	putw	toascii
atoi	hdestroy	qsort	tolower
atol	hsearch	raise	toupper
bsearch	isalnum	rand	tsearch
cfgetispeed	isalpha	rewind	twalk
cfgetospeed	isascii	scalb ²	tzset
cfsetispeed	isatty	scanf	ungetc
cfsetospeed	iscntrl	setbuf	vfprintf
clearerr	isdigit	setitimer ¹	vprintf
clock	isgraph	setjmp	vsprintf
crypt ¹	islower	setkey ¹	wcstombs
ctermid	isnan	setlabel ³	wctomb
ctime	isnand	setvbuf	
cuserid	isprint	sleep	
difftime	ispunct	sprintf	
div	isspace	srand	1-see the C library
dup2	isupper	sscanf	changes at the
encrypt ¹	isxdigit	strcat	beginning of this
fclose	labs	strchr	chapter.
fdopen ¹	ldexp	strcmp	
feof	ldiv	strcpy	2-New additions:
ferror	lfind	strcspn	Not in the SCD 2.2
fflush	localtime	strdup	
fgetc	lockf ¹	strlen	
fgetpos	logb ²	strncat	3- change of status
fgets	longjmp	strncmp	between SCD 2.2 and
fileno	lsearch	strncpy	SCD 2.3
fmtmsg	mblen	strpbrk	
fopen	mbstowcs	strrchr	
fprintf	mbtowc	strspn	
fputc	memccpy	strstr	
fputs	memchr	strtod	
fread	memcmp	strtok	
freopen	memcpy	strtol	
frexp	memmove	strtoul	
fscanf	memset	swab	
fseek	mkfifo	sysinfo ¹	
fsetpos	mktemp	tcdrain	
ftell	mktime	tcflow	
fwrite	modf	tcflush	
getc	monitor	tcgetattr	

The Network Services Library

Overview

This section contains the libnsl interfaces listed in the *System V Application Binary Interface*, and described in the *System V Interface Definition, Third Edition*.

The libnsl Interfaces

The interfaces listed below in Table 6-9 have been included in SCD 2.3 because they are REQUIRED to be present on all systems conforming to the *System V Application Binary Interface*, in the dynamic library `/usr/lib/libnsl.so.1`.

The interfaces found in Table 6-10 are also REQUIRED to be present on an ABI-conforming system. Systems without networking capabilities are not required to implement these interfaces, but must provide an entry point in libnsl for each. Entry points which are provided as stubs and not implemented must fail normally and set the external symbol `errno` to `ENOSYS`. Additionally, DEPRECATED functions needed for socket support can be found in Table 6-11

ABI Extension

The SCD requires `/usr/lib/libnsl.so.1` to have functions which are not specified by or different from the gABI. These extra functions are either not defined in the SVID, or, are defined differently in the SCD than in the SVID. The semantics for these additional/modified function definitions are available in the SCD 2.3 Interface Semantics document.

Table 6-8 Exported data for libnsl

<code>_nderror[0x4]</code>	<code>svc_fdset¹</code>	1- see libnsl	2- DEPRECATED
<code>_null_auth²</code>	<code>t_errno [0x4]</code>	library changes at	
<code>rpc_createerr[0x16]</code>		the beginning of	
		this chapter.	

Table 6-9 libnsl contents, Part 1 of 2

<code>t_accept</code>	<code>t_getstate¹</code>	<code>t_rcvrel</code>	<code>t_unbind</code>
<code>t_alloc¹</code>	<code>t_listen</code>	<code>t_rcvudata</code>	
<code>t_bind</code>	<code>t_look</code>	<code>t_rcvuderr</code>	
<code>t_close</code>	<code>t_open</code>	<code>t_snd</code>	
<code>t_connect</code>	<code>t_optmgmt</code>	<code>t_snddis</code>	1- see the libnsl
<code>t_error</code>	<code>t_rcv</code>	<code>t_sndrel</code>	library changes at
<code>t_free</code>	<code>t_rcvconnect</code>	<code>t_sndudata</code>	the beginning of
<code>t_getinfo</code>	<code>t_rcvdis</code>	<code>t_sync</code>	this chapter.

Table 6-10 libnsl contents, part 2 of 2

authdes_getucred	rpc_reg	xdr_float
authdes_seccreate	rpcb_getaddr	xdr_free
authnone_create	rpcb_getmaps	xdr_int
authsys_create	rpcb_gettime	xdr_long
authsys_create_default	rpcb_rmtcall	xdr_opaque
clnt_create	rpcb_set	xdr_opaque_auth
clnt_dg_create	rpcb_unset	xdr_pointer
clnt_pcreateerror	setnetconfig	xdr_reference
clnt_perrno	setnetpath	xdr_rejected_reply
clnt_perror	svc_create	xdr_replymsg
clnt_raw_create	svc_dg_create	xdr_short
clnt_spccreateerror	svc_fd_create	xdr_string
clnt_sperrno	svc_getreqset	xdr_u_char
clnt_sperror	svc_raw_create	xdr_u_long
clnt_tli_create	svc_reg	xdr_u_short
clnt_tp_create	svc_run	xdr_union
clnt_vc_create	svc_sendreply	xdr_vector
endnetconfig	svc_tli_create	xdr_void
endnetpath	svc_tp_create	xdr_wrapstring
freenetconfig	svc_unreg	xdrmem_create
getnetconfig	svc_vc_create	xdrrec_create
getnetconfigent	svcerr_auth	xdrrec_eof
getnetname	svcerr_decode	xdrrec_skiprecord
getnetpath	svcerr_noproc	xdrstdio_create
getpublickey	svcerr_noprog	xprt_register
getsecretkey	svcerr_progvers ¹	xprt_unregister
host2netname	svcerr_systemerr	
key_decryptsession	svcerr_weakauth	1- see libnsl library
key_encryptsession	taddr2uaddr	changes at the beginning
key_gendes	uaddr2taddr	of this chapter.
key_setsecret	user2netname	
nc_perror	xdr_accepted_reply ²	2- was dropped by mistake
netdir_free	xdr_array	in the SCD 2.2
netdir_getbyaddr	xdr_authsys_parms	
netdir_getbyname	xdr_bool	
netdir_options	xdr_bytes	
netname2host	xdr_callhdr	
netname2user	xdr_callmsg	
rpc_broadcast	xdr_char	
rpc_broadcast_exp ¹	xdr_double	
rpc_call ¹	xdr_enum	

Networking Interface Set

This section contains the socket inter-networking interface, primarily used with the TCP/IP protocol suite. This is a REQUIRED interface set. It is also a DEPRECATED interface set effective November 1st, 1993. This interface set will not be removed from the SCD before November 1st, 1996.

The description of this interface is available in the 2.3 Interface Semantics document.

All functions must be provided by systems from one or more of the libraries /usr/lib/libnsl.so.1, and /usr/lib/libsocket.so.1. Table 6-11 and Table 6-12 document which system libraries are REQUIRED to provide each of the various socket functions.

Table 6-11 Socket Functions in libnsl

gethostbyaddr	inet_addr	inet_ntoa
gethostbyname	inet_netof	

Table 6-12 Socket Functions in libsocket

accept	inet_lnaof	shutdown
bind	inet_makeaddr	socket
connect	inet_network	
getpeername	listen	
getprotobyname	recv	
getprotobyname	recvfrom	
getprotoent	recvmsg	
getservbyname	send	
getservbyport	sendmsg	
getsockname	sendto	
getsockopt	setsockopt	

Structures and Manifest Constants

The Figures 6-2 through 6-5 contain the values of manifest constants and type declarations of the data types needed for the socket functions.

Figure 6-2 Manifest Constants and Data Types from <sys/socket.h>

```

/* Types */
#define SOCK_STREAM      2          /* stream socket */
#define SOCK_DGRAM      1          /* datagram socket */
#define SOCK_RAW        4          /* raw-protocol interface */
#define SOCK_RDM        5          /* reliably-delivered message */
#define SOCK_SEQPACKET  6          /* sequenced packet stream */

/* Option flags per-socket. */
#define SO_DEBUG         0x0001    /* turn on debugging info recording */
#define SO_ACCEPTCONN   0x0002    /* socket has had listen() */
#define SO_REUSEADDR    0x0004    /* allow local address reuse */
#define SO_KEEPAVIVE    0x0008    /* keep connections alive */
#define SO_DONTROUTE    0x0010    /* just use interface addresses */
#define SO_BROADCAST    0x0020    /* permit sending of broadcast msgs */
#define SO_USELOOPBACK  0x0040    /* bypass hardware when possible */
#define SO_LINGER        0x0080    /* linger on close if data present */
#define SO_OOBNLINE     0x0100    /* leave received OOB data in line */

/* Additional options, not kept in so_options. */
#define SO_SNDBUF        0x1001    /* send buffer size */
#define SO_RCVBUF        0x1002    /* receive buffer size */
#define SO_SNDLOWAT      0x1003    /* send low-water mark */
#define SO_RCVLOWAT      0x1004    /* receive low-water mark */
#define SO_SNDTIMEO      0x1005    /* send timeout */
#define SO_RCVTIMEO      0x1006    /* receive timeout */
#define SO_ERROR         0x1007    /* get error status and clear */
#define SO_TYPE          0x1008    /* get socket type */
#define SO_PROTOCOL      0x1009    /* get/set protocol type */

/* Structure used for manipulating linger option. */
struct linger {
    int    l_onoff;          /* option on/off */
    int    l_linger;        /* linger time */
};

/* Level number for (get/set) sockopt() to apply to socket itself. */
#define SOL_SOCKET       0xffff    /* options for socket level */

/* Address families. */

#define AF_UNSPEC        0          /* unspecified */
#define AF_UNIX         1          /* local to host (pipes, portals) */
#define AF_INET         2          /* internetwork: UDP, TCP, etc. */
#define AF_IMPLINK      3          /* arpanet imp addresses */
#define AF_PUP          4          /* pup protocols: e.g. BSP */
#define AF_CHAOS        5          /* mit CHAOS protocols */
#define AF_NS           6          /* XEROX NS protocols */
#define AF_NBS          7          /* nbs protocols */
#define AF_ECMA         8          /* european computer manufacturers */
#define AF_DATAKIT      9          /* datakit protocols */
#define AF_CCITT        10         /* CCITT protocols, X.25 etc */
#define AF_SNA          11         /* IBM SNA */
#define AF_DECnet       12         /* DECnet */
#define AF_DLI          13         /* Direct data link interface */
#define AF_LAT          14         /* LAT */
#define AF_HYLINK      15         /* NSC Hyperchannel */

```

```

#define AF_APPLETALK      16          /* Apple Talk */
#define AF_NIT            17          /* Network Interface Tap */
#define AF_802            18          /* IEEE 802.2, also ISO 8802 */
#define AF_OSI            19          /* umbrella for all families used */
#define AF_X25            20          /* CCITT X.25 in particular */
#define AF_OSINET        21          /* AFI = 47, IDI = 4 */
#define AF_GOSIP          22          /* U.S. Government OSI */
#define AF_MAX            22

/* Structure used by kernel to store most addresses. */
struct sockaddr {
    u_short  sa_family;          /* address family */
    char     sa_data[14];       /* up to 14 bytes of direct address */
};

/* Structure used by kernel to pass protocol * information in raw sockets. */
struct sockproto {
    u_short  sp_family;          /* address family */
    u_short  sp_protocol;       /* protocol */
};

/* Protocol families, same as address families for now. */
#define PF_UNSPEC          AF_UNSPEC
#define PF_UNIX            AF_UNIX
#define PF_INET            AF_INET
#define PF_IMPLINK        AF_IMPLINK
#define PF_PUP             AF_PUP
#define PF_CHAOS           AF_CHAOS
#define PF_NS              AF_NS
#define PF_NBS             AF_NBS
#define PF_ECMA            AF_ECMA
#define PF_DATAKIT         AF_DATAKIT
#define PF_CCITT           AF_CCITT
#define PF_SNA             AF_SNA
#define PF_DECnet          AF_DECnet
#define PF_DLI             AF_DLI
#define PF_LAT             AF_LAT
#define PF_HYLINK          AF_HYLINK
#define PF_APPLETALK       AF_APPLETALK
#define PF_NIT             AF_NIT
#define PF_802             AF_802
#define PF_OSI             AF_OSI
#define PF_X25             AF_X25
#define PF_OSINET          AF_OSINET
#define PF_GOSIP           AF_GOSIP
#define PF_MAX             AF_MAX

/* Maximum queue length specifiable by listen. */
#define SOMAXCONN          5

/* Message header for recvmsg and sendmsg calls. */
struct msghdr {
    caddr_t    msg_name;          /* optional address */
    int        msg_namelen;       /* size of address */
    struct iovec *msg_iov;        /* scatter/gather array */
    int        msg_iovlen;        /* # elements in msg_iov */
    caddr_t    msg_accrights;     /* access rights sent/received */
    int        msg_accrightslen;

```

```
};

#define MSG_OOB          0x1          /* process out-of-band data */
#define MSG_PEEK         0x2          /* peek at incoming message */
#define MSG_DONTROUTE    0x4          /* send without using routing tables */
#define MSG_MAXIOVLEN    16

/* option header */
struct opthdr {
    long    level;          /* protocol level affected */
    long    name;           /* option to modify */
    long    len;            /* length of option value */
};

#define OPTLEN(x)          (((x) + sizeof (long) - 1) / sizeof (long)) * sizeof (long)
#define OPTVAL(opt)        ((char *) (opt + 1))

/* the optdefault structure is used for internal tables of option default values. */
struct optdefault {
    int     optname;        /* the option */
    char    *val;           /* ptr to default value */
    int     len;            /* length of value */
};

/* the opproc structure is used to build tables of options processing functions for
dooptions(). */
struct opproc {
    int     level;          /* options level this function handles */
    int     (*func)();      /* the function */
};

/* This structure is used to encode pseudo system calls */
struct socksreq {int     args[7];};

/* This structure is used for adding new protocols to the list supported by sockets. */
struct socknewproto {
    int     family;         /* address family (AF_INET, etc.) */
    int     type;           /* protocol type (SOCK_STREAM, etc.) */
    int     proto;          /* per family proto number */
    dev_t   dev;            /* major/minor to use (must be a clone) */
    int     flags;          /* protosw flags */
};
```


Figure 6-3 Manifest Constants and Data Types from <netinet/in.h>

```

/* IP address */
struct in_addr {
    union {
        struct {u_char s_b1, s_b2, s_b3, s_b4;} S_un_b;
        struct {u_short s_w1, s_w2;} S_un_w;
        u_long S_addr;
    } S_un;
};

/* socket address using IP */

struct sockaddr_in {
    short sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

/* Options for use with [gs]etsockopt at the IP level. */

#define IP_OPTIONS 1 /* set/get IP per-packet options */
#define IP_HDRINCL 2 /* int; header is included with data (raw) */
#define IP_TOS 3 /* int; IP type of service and precedence */
#define IP_TTL 4 /* int; IP time to live */
#define IP_RECVOPTS 5 /* bool; receive all IP options w/datagram */
#define IP_RECVRETOPTS 6 /* bool; receive IP options for response */
#define IP_RECVDSTADDR 7 /* bool; receive IP dst addr w/datagram */
#define IP_RETOPTS 8 /* ip_opts; set/get IP per-packet options */

#define IP_MULTICAST_IF 0x10 /* set/get IP multicast interface */
#define IP_MULTICAST_TTL 0x11 /* set/get IP multicast timetolive */
#define IP_MULTICAST_LOOP 0x12 /* set/get IP multicast loopback */
#define IP_ADD_MEMBERSHIP 0x13 /* add an IP group membership */
#define IP_DROP_MEMBERSHIP 0x14 /* drop an IP group membership */

#define IP_DEFAULT_MULTICAST_TTL 1 /* normally limit m'casts to 1 hop */
#define IP_DEFAULT_MULTICAST_LOOP 1 /* normally hear sends if a member */

/* Argument structure for IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP. */

struct ip_mreq {
    struct in_addr imr_multiaddr; /* IP multicast address of group */
    struct in_addr imr_interface; /* local IP address of interface */
};

```

Figure 6-4 Manifest Constants and Data Types from <netdb.h>

```
struct hostent {
    char    *h_name;           /* official name of host */
    char    **h_aliases;       /* alias list */
    int      h_addrtype;        /* host address type */
    int      h_length;          /* length of address */
    char    **h_addr_list;     /* list of addresses from name server */
#define     h_addr             h_addr_list[0] /* address, for backward compatibility */
};

struct servent {
    char    *s_name;           /* official service name */
    char    **s_aliases;       /* alias list */
    int      s_port;            /* port # */
    char    *s_proto;           /* protocol to use */
};

struct protoent {
    char    *p_name;           /* official protocol name */
    char    **p_aliases;       /* alias list */
    int      p_proto;           /* protocol # */
};

#define HOST_NOT_FOUND        1      /* Authoritative Answer Host not found */
#define TRY_AGAIN              2      /* Non-Auth. Host not found, or SERVERFAIL */
#define NO_RECOVERY            3      /* Non recover errors:FORMERR,REFUSED,NOTIMP */
#define NO_DATA                4      /* Valid name, no data rec. of requested type */
#define NO_ADDRESS             NO_DATA /* no address, look for MX record */
```

Figure 6-5 Manifest Constants and Data Types from <errno.h>

```
#define EADDRINUSE            125      /* Address already in use */
#define EADDRNOTAVAIL         126      /* Can't assign requested address */
#define EAFNOSUPPORT          124      /* Address family not supported by */
#define EALREADY              149      /* operation already in progress */
#define ECONNREFUSED          146      /* Connection refused */
#define EINPROGRESS           150      /* operation now in progress */
#define EISCONN               133      /* Socket is already connected */
#define EMSGSIZE              97       /* Message too long */
#define ENETUNREACH           128      /* Network is unreachable */
#define ENOTCONN              134      /* Socket is not connected */
#define ENOTSOCK              95       /* Socket operation on non-socket */
#define EOPNOTSUPP            122      /* Operation not supported on socket */
#define EPROTONOSUPPORT       120      /* Protocol not supported */
#define EPROTOTYPE            98       /* Protocol wrong type for socket */
#define ETIMEDOUT             145      /* Connection timed out */

#define EWOULDBLOCK           EAGAIN
```

Dynamic Object File Loading

Overview

The run-time dynamic linking facilities of the system are made available to the executing application program through the functions in `libdl` as shown in Table 6-13. This is a REQUIRED interface set in the library `/usr/lib/libdl.so.1`.

Table 6-13 libdl contents

<code>dlclose</code>	<code>dLError</code>	<code>dlopen</code>	<code>dlsym</code>
----------------------	----------------------	---------------------	--------------------

The particulars on dynamic linking and loading, path name resolution, data initialization functions, symbol relocation and binding, and automatic loading of secondary objects are given in Chapter 5 of this document and in the normative documents it references, the *System V Application Binary Interface* and the *System V Application Binary Interface, SPARC Processor Supplement*.

ABI Extensions

The SCD requires an additional library `/usr/lib/libdl.so.1` which is not specified by the gABI. This library contains the functions `dlclose`, `dLError`, `dlopen` and `dlsym`. These extra functions are not defined in the SVID. The semantics for these additional function definitions are available in the SCD 2.3 Interface Semantics document.

Figure 6-6 contains manifest constants for `dlopen`.

Figure 6-6 Manifest Constants and Data Types from `<dlfcn.h>`

```
/* Valid values for mode argument to dlopen. */

#define RTLD_LAZY      1      /* lazy function call binding */
#define RTLD_NOW      2      /* immediate function call binding */
```

Multithreading Library

Overview

The services specified in this section provide applications with the ability to create multiple “threads of control” within their address spaces. The interface set described here resides entirely in the library `/usr/lib/libthread.so.1`, and represents a REQUIRED INTERFACE. The library contains the entry points described in the table below.

Table 6-14 libthread contents

<code>cond_broadcast</code>	<code>rw_rdlock</code>	<code>setcontext</code>	<code>thr_keycreate</code>
<code>cond_destroy</code>	<code>rw_tryrdlock</code>	<code>sigaction</code>	<code>thr_kill</code>
<code>cond_init</code>	<code>rw_trywrlock</code>	<code>sigprocmask</code>	<code>thr_main</code>
<code>cond_signal</code>	<code>rw_unlock</code>	<code>sigwait</code> ¹	<code>thr_min_stack</code>
<code>cond_timedwait</code>	<code>rw_wrlock</code>	<code>sleep</code>	<code>thr_self</code>
<code>cond_wait</code>	<code>rwlock_destroy</code>	<code>thr_continue</code>	<code>thr_setconcurrency</code>
<code>fork1</code>	<code>rwlock_init</code>	<code>thr_create</code>	<code>thr_setprio</code>
<code>mutex_destroy</code>	<code>sema_destroy</code>	<code>thr_exit</code>	<code>thr_setspecific</code>
<code>mutex_init</code>	<code>sema_init</code>	<code>thr_getconcurrency</code>	<code>thr_sigsetmask</code>
<code>mutex_lock</code>	<code>sema_post</code>	<code>thr_getprio</code>	<code>thr_suspend</code>
<code>mutex_trylock</code>	<code>sema_trywait</code>	<code>thr_getspecific</code>	<code>thr_yield</code>
<code>mutex_unlock</code>	<code>sema_wait</code>	<code>thr_join</code>	

¹ - This function is designated EXPERIMENTAL because the SCD definition differ from that in POSIX. There exist implementations which conform to this definition, and others which conform to POSIX. The future direction of the SCD is full conformance with POSIX (see the SCD-IS for more information):

SCD:	int	<code>sigwait (sigset_t *setp);</code>
POSIX:	int	<code>sigwait (const sigset_t *setp, int *signo);</code>

For an application to use libthread correctly, it must specify a reference to libthread in a DT_NEEDED entry prior to a DT_NEEDED entry which specifies either libsys or libc. libthread redefines the semantics of a number of entry points normally provided by either libsys or libc, primarily to support the correct management of signals in a multithreaded program.

Additional Interfaces in libsys/libc

When a system provides the multithreading interface specified in this section, it also provides enhancements to the system services provided by libsys and libc. These enhancements are, for the most part, transparent to the application. However, in a few cases, these enhancements manifest themselves as new entry points in these libraries. This is necessary as the nature of the manner in which the interface was previously defined made it impossible to express a multi-threaded-safe implementation. For the most part, these new routines have the name of the old routine with the string `_r` appended. All of these entry points are part of libc. Those whose “root names” are also present in libsys also reside there. All, with the exception of `__errno`, have synonyms. The entry points so added are described in Table 6-16, “Additions to libsys,” on page 6-25 and Table 6-16, “Additions to libsys,” on page 6-25.

ABI Extensions

The SCD requires an additional library `/usr/lib/libthread.so.1` which is not specified by the gABI. The functions defined in `/usr/lib/libthread.so.1` are not defined in the SVID. The semantics for these additional function definitions are available in the SCD 2.3 Interface Semantics document.

Table 6-15 Additions to-libc

__errno	funlockfile	localtime_r	strtok_r
asctime_r ¹	getc_unlocked	putc_unlocked	
ctime_r ¹	getchar_unlocked	putchar_unlocked	
flockfile	gmtime_r	rand_r	

1 - These functions are designated EXPERIMENTAL because the SCD definitions differ from those in POSIX 1003.1c. There exist implementations which conform to these definitions, and others which conform to POSIX. The future direction of the SCD is full conformance with POSIX. (see the SCD-IS for more information):

SCD:	char	*asctime_r(const struct tm *tm, char *buf, int buflen);
POSIX:	char	*asctime_r(const struct tm *tm, char *buf);
SCD:	char	*ctime_r(const time_t *clock, char *buf, int buflen);
POSIX:	char	*ctime_r(const time_t *clock, char *buf);

Table 6-16 Additions to libsys

fgetgrent_r	getgrgid_r ¹	getpwent_r	readdir_r ¹
fgetpwent_r	getgrnam_r ¹	getpwnam_r ¹	ttynname_r ¹
getgrent_r	getlogin_r ¹	getpwuid_r ¹	

1 - These functions are designated EXPERIMENTAL because the SCD definitions differ from those in POSIX 1003.1c. There exist implementations which conform to these definitions, and others which conform to POSIX. The future direction of the SCD is full conformance with POSIX (see the SCD-IS for more information):

SCD:	struct group	*getgrgid_r (gid_t gid, struct group *result, char *buffer, int buflen);
POSIX:	int	getgrgid_r (gid_t gid, struct group *grp, char *buffer, size_t bufsz, struct group **result);
SCD:	struct group	*getgrnam_r (const char *name, struct group *result, char *buffer, int buflen);
POSIX:	int	getgrnam_r (const char *name, struct group *grp, char *buffer, size_t bufsz, struct group **result);
SCD:	char	*getlogin_r (char *name, int namelen);
POSIX:	int	getlogin_r (char *name, size_t namelen);
SCD:	struct passwd	*getpwnam_r (const char *name, struct passwd *result, char *buffer, int buflen);
POSIX:	int	getpwnam_r (const char *name, struct passwd *pwd, char *buffer, size_t bufsz, struct passwd **result);
SCD:	struct passwd	*getpwuid_r (uid_t uid, struct passwd *result, char *buffer, int buflen);
POSIX:	int	getpwuid_r (uid_t uid, struct passwd *pwd, char *buffer, size_t bufsz, struct passwd **result);
SCD:	struct dirent	*readdir_r (DIR *dirp, struct dirent *res);
POSIX:	int	readdir_r (DIR *dirp, struct dirent *entry, struct dirent **result);
SCD:	char	*ttynname_r (int fd, char *buf, int len);
POSIX:	int	ttynname_r (int fd, char *name, size_t namesz);

Table 6-17 modified libsys functions

fork¹

1- fork multi-threading features are designated as EXPERIMENTAL, because the SCD definition differs from that in POSIX 1003.1c. When fork is called, all threads of the process get forked, However in POSIX only a single thread is forked (similar to the SCD fork1).

Figure 6-7 Manifest Constants and Data Types from <synch.h>

```
#define USYNC_THREAD    0                /* private to a process */
#define USYNC_PROCESS   1                /* shared by processes */

typedef struct { /* unspecified, but sizeof(mutex_t) is 24. */} mutex_t;
typedef struct { /* unspecified, but sizeof(cond_t) is 16. */} cond_t;
typedef struct { /* unspecified, but sizeof(sema_t) is 48. */} sema_t;
typedef struct { /* unspecified, but sizeof(rwlock_t) is 64. */} rwlock_t;
```

Figure 6-8 Manifest Constants and Data Types from <thread.h>

```
typedef      unsigned int  thread_t;
typedef      unsigned int  thread_key_t;

#define THR_BOUND          0x00000001
#define THR_NEW_LWP        0x00000002
#define THR_DETACHED       0x00000040
#define THR_SUSPENDED      0x00000080
#define THR_DAEMON         0x00000100
```

Figure 6-9 Manifest Constants and Data Types from <errno.h>

```
/* When _REENTRANT is defined, a multithreaded application is being constructed. */
#ifdef _REENTRANT
#define errno (*__errno())
#else
extern int errno;
#endif
```

Asynchronous I/O

Overview

The services specified in this section provide applications with the ability to invoke a number of file operations asynchronously with the execution of the application program. The interface set described here resides entirely in the library `/usr/lib/libaio.so.1`, and represents a REQUIRED and DEPRECATED INTERFACE. The library contains the entry points described in the table below.

ABI Extensions

The SCD requires an additional library `/usr/lib/libaio.so.1` which is not specified by the gABI. The functions defined in `/usr/lib/libaio.so.1` are not defined in the SVID. The semantics for these additional function definitions are available in the SCD 2.3 Interface Semantics document.

Table 6-18 libaio Contents

<code>aiocancel</code>	<code>aioread</code>	<code>aiowait</code>	<code>aiowrite</code>
------------------------	----------------------	----------------------	-----------------------

Figure 6-10 Manifest Constants and Data Types from `<sys/async.h>`

```
#define AIO_INPROGRESS -2      /* values not set by the system */

typedef struct aio_result_t {
    int    aio_return;          /* return value of read or write */
    int    aio_errno;          /* errno generated by the IO */
} aio_result_t;
```

The Math Library

Overview

The math library, libm, contains several mathematical functions listed in the table “libm Contents” below. These interfaces are defined in the System V Interface Definition, Third Edition. The math library is a REQUIRED interface set with reference name /usr/lib/libm.so.1.

Table 6-19 libm contents

acos	erf	lgamma	y1
acosh	erfc	log10	yn
asin	exp	log	
asinh	fabs	pow	
atan	floor	remainder	
atan2	fmod	sin	
atanh	gamma	sinh	
cbrt	hypot	sqrt	
ceil	j0	tan	
cos	j1	tanh	
cosh	jn	y0	

The Large files Library

Overview

The operating system service routines specified in this section provide access to large files, large file systems, and associated resources. Their semantics are the same as the corresponding routines without the `lf_` prefixes, except as noted below. All of the `lf_` interfaces reside in the `liblf` library `/usr/lib/liblf.so.1` regardless of whether the corresponding small-file interfaces reside in `libsys` or `libc`. None has a synonym (identical interface with `'_'` prefix.) Since this is a new interface set, it is included in the SCD as an EXPERIMENTAL INTERFACE. Table 6-20 lists the new routines provided in this interface set. Figure 6-11 through Figure 6-15 list the new data structures and manifest constants required to support this interface set.

Table 6-20 liblf contents

<code>lf_fcntl</code>	<code>lf_fstatvfs</code>	<code>lf_lstat</code>	<code>lf_stat</code>
<code>lf_fpathconf</code>	<code>lf_ftell</code>	<code>lf_mmap</code>	<code>lf_statvfs</code>
<code>lf_fseek</code>	<code>lf_getrlimit</code>	<code>lf_pathconf</code>	<code>lf_tell</code>
<code>lf_fstat</code>	<code>lf_lseek</code>	<code>lf_setrlimit</code>	

FUTURE DIRECTION:

At the time of the writing of this document, members of the Unix community are meeting in an attempt to agree on a standard implementation of 64-bit files on a 32-bit system. Members of SPARC International are taking part in these discussions. If agreement on these interfaces is reached by the industry, the interfaces for large file support contained in this section will be modified, if necessary, to match the agreed to interfaces.

Figure 6-11 Data Types Defined in `<sys/fcntl.h>`

```
typedef struct lf_flock {
    short      l_type;
    short      l_whence;
    int        l_pad_1;
    lf_off_t   l_start;
    lf_off_t   l_len;
    long       l_sysid;
    pid_t      l_pid;
    long       pad[4];
} lf_flock_t;
```

Figure 6-12 Data Types Defined in <sys/stat.h>

```
typedef struct lf_stat {
    dev_t      st_dev;
    long       st_pad1[3];
    ino_t      st_ino;
    mode_t     st_mode;
    nlink_t    st_nlink;
    uid_t      st_uid;
    gid_t      st_gid;
    dev_t      st_rdev;
    long       st_pad2[2];
    lf_off_t   st_size;
    long       st_pad3;
    timestruc_t st_atim;
    timestruc_t st_mtim;
    timestruc_t st_ctim;
    long       st_blksize;
    int64_t     st_blocks;
    char       st_fstype[_ST_FSTYPSZ];
    long       st_pad4[8];
} lf_stat_t;
```

Figure 6-13 Data Types Defined in <sys/resource.h>

```
struct lf_rlimit {
    lf_rlimit_t    rlim_cur;
    lf_rlimit_t    rlim_max;
};
```

Figure 6-14 Data Types Defined in <sys/statvfs.h>

```
typedef struct lf_statvfs {
    unsigned long    f_bsize;
    unsigned long    f_frsize;
    uint64_t         f_blocks;
    uint64_t         f_bfree;
    uint64_t         f_bavail;
    unsigned long    f_files;
    unsigned long    f_ffree;
    unsigned long    f_favail;
    unsigned long    f_fsid;
    char             f_basetype[FSTYPESZ];
    unsigned long    f_flag;
    unsigned long    f_namemax;
    char             f_fstr[32];
    unsigned long    f_filler[16];
} lf_statvfs_t;
```

Figure 6-15 Manifest Constants Defined in <unistd.h>

```
#define _PC_MAX_FILE_SIZE    10
```


CHAPTER 7: Formats and Protocols

SCD
2.3

Formats and Protocols

Archive file formats, networking protocols, and the terminfo data base format may be found in Chapter 7 of the *System V Application Binary Interface*.

Formats and Protocols Changes

The following are changes to the *System V ABI*, the *System V ABI SPARC Processor Supplement*, and the *System V Interface Definition* as reported to SPARC International.

#	Facility	Location	Description
1	rpcbind Operation	gABI	Change - page 7-38-The reference to IP in the first paragraph is ambiguous -- port 111 is used for IP-carried transports (rather than IP itself).

Interconnecting SCD Conforming Systems

Overview

This section contains the REQUIRED internetworking interfaces available to applications running on an SCD conforming system. Note that the networking ABI is defined by the TLI interfaces described in section BA_LIB of the *System V Interface Definition (Third Edition), Volume I*. This chapter adds to that definition by specifying that there shall be present in all SCD complying systems an Internet Protocol Suite (IPS) transport provider that is accessible through TLI. Also added are the commands, which exist in `/usr/bin`, and their associated daemons, which exist in `/usr/sbin`.

Transport Providers

All SPARC-compliant systems will provide a transport provider interface for each of the IP protocols, TCP, UDP, ICMP, and ARP. The device names for these transport provider interfaces must be `/dev/tcp`, `/dev/udp`, `/dev/icmp`, and `/dev/arp` respectively. Additionally, shared objects will be present to convert IP format universal addresses into the necessary internal format needed by the TLI interfaces. These interfaces are previously defined in the Network Services Library portion of this chapter.

Additional Interfaces

The interfaces listed below in Table 7-1 show the additional commands, protocols, and service daemons that are included to ensure inter-operability between SCD conforming systems. The table includes three columns, the command name which is invoked, the RFC number for the protocol specification as maintained by the Internet Engineering Task Force, and a short description of the feature provided.

Table 7-2 shows the “well-known” port numbers as derived from RFC 1700 that SCD conforming systems are REQUIRED to provide for supported services.

Table 7-1: Required Commands

Command	RFC	Description
rlogin	BSDNET	Remote terminal services (BSD)
rsh	BSDNET	Remote user shell (BSD)
rcp	BSDNET	Remote file copy (BSD)
rwho	BSDNET	Remote user information service (BSD)
rdate	BSDNET	Remote uptime statistics (BSD)
talk	BSDNET	Remote chat utility (BSD)
finger	rfc1288	Information server for logged on users
telnet	<many>	Interactive terminal services
ftp	rfc959	File transfer protocol
arp	rfc826	Address Resolution Protocol

Table 7-2: Well-Known Port Numbers

Keyword	Description	TCP Port Number	UDP Port Number
tcpmux	rfc1078	1	
echo	Echo	7	7
discard	Discard	9	9
systat	Active Users	11	11
daytime	Daytime	13	13
netstat	Who is up or NETSTAT	15	15
chargen	Character Generator	19	19
ftp-data	File Transfer Protocol (Data)	20	
ftp	File Transfer Protocol	21	
telnet	Terminal Connection	23	
smtp	Simple Mail Transport Protocol	25	
time	Time	37	37
name	Host Name Server	42	42
nicname	Who Is	43	43
domain	Domain Name Server	53	53
tftp	Trivial File Transfer	69	69
	Any private RJE service	77	77
finger	Finger	79	79
supdup	SUPDUP Protocol	95	
hostname	NIC Host Name Server	101	
iso-tsap	ISO-TSAP	102	
uucp-path	UUCP Path Service	117	
ntp	Network Time Protocol	123	123
x	X Window Service	6000+Display Number	

CHAPTER 8: System Commands

SCD
2.3

System Commands

Overview

This chapter contains the commands for application programs as listed in the *System V Application Binary Interface (Third Edition)*, and described in the *System V Interface Definition, (Third Edition)*.

Table 8-1. Commands for Application Programs

ar ²	false	pwd*	uucp
awk	find ¹	rm	uulog
basename ²	fmtmsg	rmdir	uustat
cat	gencat ²	sed	uux
cd†	gettxt	sh ¹	vi
chgrp	grep	sleep	wait†
chmod	id	sort	wc ²
chown	kill	stty	who ¹
cmp	line	su	
cp	ln	sum ²	
cpio	logname	tail	
compress ²	lp	tee	
date	ls	test*	
dirname ²	make ²	touch	
dd	mkdir	tr	
df	mv	true	
echo*	passwd	tty	
ed	pg	umask†	
ex ¹	pr ¹	uname	
expr	priocntl	uncompress ²	

* These commands marked are also built into the standard UNIX system shell, sh.

† These commands are only available as commands built-in to the UNIX system shell, sh.

1- see system commands changes below 2- New Additions: not in SCD 2.2

System Commands Changes

The following are changes to the basic system commands (detailed in the *System V Application Binary Interface*), as reported to SPARC International.

#	Facility	Location	Description
1	ex(BU_CMD)	SVID, Vol. 2	Change - The SVID states that the “ed compatible” option of ex causes the g suffix on substitute commands to be remembered, and toggled by repeating the suffix. Omitted from this description is the fact that this behavior is applicable only to the “&” form of substitute commands.

System Commands Changes (continued)

#	Facility	Location	Description
1	ex(BU_CMD)	SVID, Vol. 2	Change - The “c” command should be defined as “Enters input mode; the input text replaces the specified lines. The last input line becomes the current line; if no lines are input the line before the deleted line(s) becomes the current line.”
1	ex(BU_CMD)	SVID, Vol. 2	Change - The “m” command description must be changed to note that the current line becomes the last of the moved lines, rather than the first.
2	find(BU_CMD)	SVID, Vol. 2	Change the descriptions of -atime, -mtime, and -ctime from “in n days” to “n days ago.”
3	pr(BU_CMD)	SVID, Vol. 2	Change - The SVID says that using -m with the -column option will cause the -m option to override the -column option. This does not match current practice; using these two options together will be treated as an error.
3	pr(BU_CMD)	SVID, Vol. 2	Change - Comments about truncating lines in the text of the description and in the options are incorrect with respect to single column output: existing practice and P1003.2 is that truncation is not applied to single column output. The “note” in the description of the -w option is to be applicable to multi-column output only. In the description change the second paragraph to read: “By default, in multi-column mode, columns are....”
4	sh(BU_CMD)	SVID, Vol. 2	Changed - In the section marked “Input/Output” the description of “<<[-]word” states: “... \ must be used to quote the characters \, \$, ‘, and the first character of word” should be changed to read “... \ must be used to quote the characters \, \$, and ‘”. This matches both existing practice and P1003.2.
5	who(AU_CMD)	SVID, Vol. 2	<p>Change the description of the -T and -a option to “The -T and -a options to who are unspecified and cannot be relied on to be portable.”</p> <p><i>Rationale:</i></p> <p>On investigation, these options were found to differ on various SPARC implementations. The -a option is an aggregate option; rather than using this option, for SCD 2 portability an application should use the specific individual options to who that the application requires. Rather than using the -T option, an application should use either the -s or -u option for SCD 2 portability.</p>

CHAPTER 9: Execution Environment

SCD
2.3

Execution Environment

All information regarding File System Structure and Contents may be found in Chapter 9 of the *System V Application Binary Interface (Third Edition)*.

Execution Environment Changes

The following are changes to the *System V Application Binary Interface (Third Edition)*, the *System V Application Binary Interface - SPARC Processor Supplement (Third Edition)*, and the *System V Interface Definition (Third Edition)* as reported to SPARC International.

#	Facility	Location	Description
1	Root subtree - /dev	gABI	Change Figure 9-1 page 9-4, Required device files are: /dev/tty, /dev/null, /dev/console, /dev/zero. The device /dev/lpX may not be exist, as well as sub-directories /dev/rmt and /dev/mt. The definition of /dev/zero can be found in the SCD 2.3 Interface Semantics.

