

IEEE Guide to the POSIX[®] Open System Environment (OSE)

Sponsor
**Portable Applications Standards Committee
of the
IEEE Computer Society**

Approved May 2, 1995
IEEE Standards Board

Abstract: This guide presents an overview of open system concepts and their applications. Information is provided to persons evaluating systems based on the existence of, and interrelationships among, application software standards, with the objective of enabling application portability and system interoperability. A framework is presented that identifies key information system interfaces involved in application portability and system interoperability and describes the services offered across these interfaces. Standards or standards activities associated with the services are identified where they exist or are in progress. Gaps are identified where POSIX[®] Open System Environment services are not currently being addressed by formal standards. Finally, the concept of a profile is discussed with examples from several application domains.

Keywords: application portability, application interoperability, open system environments, profiles, POSIX[®]

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1995 by the Institute of Electrical and Electronics Engineers, Inc.

All rights reserved. Published 1995.

Printed in the United States of America.

ISBN 1-55937-531-0

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Introduction

[This introduction is not a normative part of IEEE Std 1003.0-1995, IEEE Guide to the POSIX Open System Environment (OSE), but is included for information only.]

Purpose

This guide describes the POSIX Open System Environment (POSIX OSE). It is intended to be used by anyone interested in using standards to construct an information processing system, including consumers, systems integrators, application developers, systems providers, and procurement agencies.

The scope of this guide is much broader than a single standard. This guide identifies standards from many different areas produced by many different organizations. The POSIX OSE is intended to be broad enough to cover the entire scope of general-purpose information processing systems. While the intent of this guide is to identify completely the user services for a general-purpose information processing system, it is acknowledged that this will take some time, and this version of the guide may be incomplete in areas that are still evolving.

It is important to note that this guide is not a base standard itself; it merely identifies standards that might be used when constructing a complete information processing system.

It is not appropriate to claim conformance to this guide because it contains no mandatory requirements. This guide is intended to be used only as a source of reference material.

Although this guide is a product of the IEEE POSIX standardization efforts, its scope is much broader than those efforts. IEEE POSIX is currently developing base standards and standardized profiles focused primarily on application programming interfaces. At the end of the introduction is a cross-reference of the POSIX standardization efforts and where they fit into the POSIX OSE. For a more detailed discussion of POSIX profiling projects, see Section 7.

The process of selecting standards for a particular application domain is called profiling. Recommendations for the production of different types of profiles are included in this guide.

It may never be necessary to implement an information processing system that provides an implementation of every standard in the POSIX OSE.

In addition to listing and categorizing existing standards efforts, this guide identifies important services that standards have not yet addressed. In areas where these services are not addressed, emerging standards efforts and existing public specifications are described. These emerging standards and public specifications are not part of the POSIX OSE. They are included in this guide to identify some of the existing work that has been done in areas that are gaps in the POSIX OSE. This guide does not promote the use of these specifications that are outside the POSIX OSE. They are included for information purposes only.

User needs and standards to meet those services are continuously expanding. As such, this guide will need regular revision to incorporate new user services and the new standards that evolve to meet those user needs.

The POSIX OSE Reference Model

To describe the POSIX OSE, this guide develops a reference model used to classify information processing standards. The reference model categorizes standards at two types of interfaces:

Application Program Interface (API) Standards

These standards govern how application software interacts with the computer system. These standards affect application portability.

External Environment Interface (EEI) Standards

These standards affect how an information processing system interacts with its external environment. These standards affect system interoperability, user interface usability, and data portability.

These standards allow users to procure portions of their information processing systems independently from multiple vendors according to the needs of each user.

The services provided at the interfaces are classified into four major categories:

- System services
- Communication services
- Information services
- Human/Computer interaction services

Within these categories, service component areas are identified.

Using the reference model, a general set of services for each component area is developed. For each of the services, existing or emerging standards are identified that address each of the services. If a service is not completely addressed by an existing or emerging standard, this gap in the standards is noted.

Goals

The POSIX OSE described in this guide should provide services to satisfy the following objectives, summarized from 3.1.

Application Portability at the Source Code Level

To allow for movement of source code and data to a variety of application platforms

System Interoperability

To allow application software and application platform interoperability

User Portability

To allow people to use a wide range of application platforms without retraining

Accommodation of Standards

To provide users and vendors with information about key interface specifications related to OSE objectives

Accommodation of New Information System Technology

To allow for migration to new technologies and a variety of marketplace solutions

Application Platform Scalability

To allow portability and software reuse across application platform types

Distributed System Scalability

To assure that related standards do not inappropriately limit the growth of distributed systems

Implementation Transparency

To allow the widest latitude in providing consistent and standard interfaces to the application, regardless of the underlying implementation technology

Functional Requirements of the User

To allow clear statement of user needs and provide context for identifying related standards

Benefits

The following items are some of the benefits derived from the use of POSIX OSE.

Integration of Components From Multiple Vendors

As the standards for system integration and system interoperability are produced and implemented, users will have the choice of mixing software and equipment from multiple vendors. This will allow users to tailor their information processing system to their particular needs by selecting hardware and software based on the needs of the application rather than the ability of the hardware and software to interoperate with the existing equipment.

Efficient Development and Implementation

Normally, systems users and providers have development and implementation activities that utilize personnel possessing skills in a specific computer environment. As a result of this specialization, a change in the target computer environment for a developer requires significant retraining expense. AS standards for application portability, system interoperability, and system integration are developed, computer personnel will begin to develop skills in working with these standards.

This will allow a company to hire personnel with existing skills that can be put to use in their operation. In addition, within a company, resources can be redeployed between development efforts with a minimum of retraining.

Efficient Porting of Applications

The difficulty of moving an application from one hardware or software environment to another is widely known. The porting of an application that uses standards-based interfaces to another system that provides the same standards-based interfaces is considerably simpler than ports involving completely different systems. The amount of system tailoring (i.e., changes to either the operating or application system required to make them work well together) is greatly reduced.

Related Standards Activities

In addition to this guide, the Portable Applications Standards Committee (PASC) has authorized other standards activities that are related to the content of this guide.

The following table summarizes the current PASC standardization efforts¹ and how they relate to sections of this guide:

Project	Standard/Profile	Section
P1003.1, .1a	System Interfaces	4.2
P1003.1b, .1d	Realtime (formerly P1003.4)	4.2
P1003.1c	Threads (formerly P1003.4)	4.2
P1003.1e	Security API (formerly P1003.6.1)	5.2
P1003.1f	Transparent File Access (formerly P1003.8)	4.3
P1003.1g	Protocol-Independent Network API (formerly P1003.12)	4.3
P1003.2, .2b	Shell and Utilities	4.7
P1003.2c	Security Utilities (formerly P1003.6.2)	5.2
P1003.2d	Batch Queueing Extensions	4.7
P1003.5	Ada Bindings	4.1
P1003.5b	Ada Realtime Binding (formerly P1003.20)	4.1
P1003.9	Fortran Bindings	4.1
P1003.10	Supercomputing Profile	7.2
P1003.13	Realtime Profile	7.2
P1003.14	Multiprocessing Profile	7.2
P1003.18	POSIX Platform Profile	7.2
P1003.21	Realtime Distributed Systems Communications	4.3
P1003.22	Guide to POSIX OSE Security Framework	5.2
P1201.1	Uniform API for Graphical User Interfaces	4.9
P1201.2	User Interface Drivability	4.9
P1224	OSI API — Abstract Data Manipulation	4.3
P1224.1	OSI API — X.400 Electronic Mail/Messaging	4.3
P1224.2	OSI API — X.500 Directory Services (formerly P1003.17)	4.3
P1238.0	OSI API Common Support Functions	4.3
P1238.1	OSI API FTAM Test Methods and C Binding	4.3
P1327	OSI API Abstract Data Manipulation — C Binding	4.3
P1327.1	OSI API X.400 — C Binding	4.3
P1327.2	OSI API X.500 — C Binding	4.3
P1387. <i>n</i>	System Administration (formerly P1003.7. <i>n</i>)	5.3
P2003. <i>n</i>	Test Methods (formerly P1003.3. <i>n</i>)	

Most of these efforts are in the areas of API standards and standardized profiles.

¹A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903, USA; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

Extensions are approved as “amendments” or “revisions” to this document, following IEEE and ISO/IEC procedures.

Approved amendments are published separately until the full document is reprinted and such amendments are incorporated in their proper positions.

If you have an interest in participating in the PASC working groups addressing these issues, please send your name, address, and phone number to the Secretary, IEEE Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA, and ask to have this forwarded to the chairperson of the appropriate PASC working group. If you have an interest in participating in this work at the international level, contact your ISO/IEC national body.

IEEE Std 1003.0-1995 was prepared by the IEEE P1003.0 working group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the P1003.0 working group was as follows:

Portable Applications Standards Committee

Chair:	Lowell Johnson
Vice-Chairs:	Jay Ashford
	Andrew Josey
	Barry Needham
	Charles Severance
	Jon Spencer
Secretary:	Charles Severance
Treasurer:	Peter Smith

1003.0 Working Group Officials

Chair:	Allen Hankinson (1987–1993)
	Kevin Lewis (1993–1995)
Vice-Chair:	Kevin Lewis (1987–1993)
	Fritz Schulz (1993–1995)
Production Editor:	Hal Jespersen
Technical Editor:	Fritz Schulz
Secretary:	Charles Severance

Working Group

Michael Aaby
Michelle Aden
Bill Allcorn
Gary Andrews
Bengt Asker
Jeanne Baccash
Jayne Baker
Rick Barbour
Ralph Barker
Tony Barrese
Jon Becker
Erwin R. Bender
Rich Bergman
Andy Bihain
Lorenzo Bonanni
Kevin Brady
Steve Brooks
Steve Carpenter
Tim Carter
David Chinn
J. J. Cinecoe
Michel Colin
Bud Conrad
Art Corey
Jean-Michel Cornu
Joe Cote
Bernard Cox
Elizabeth Crouse
Francis Deckelman
Shane Deichman
Simion Diky
Dave Dodge
Dominic Dunlop
Mat Einseln
Dave Febrache
Donna Fisher
Don Folland
David Folsom
Kester Fong
Rick Forberg
Thomas Ford
Bob Gambrel

Daniel Green
Quin Hahn
Dale Harris
John Hill
Richard Holbert
Michel Howard
Terry Humphrey
Jeff Hustad
E. Lee Hutchins
Jim Isaak
Clariest Iselt
Petr Janeczek
Michael Jende
James Johnson
Lorraine Kevra
Walter C. Keyser
Michael Kjolsrud
Bob Knighten
Bob Kruger
Mike Lambert
Doris Lebovits
Kevin Leininger
W. Edward Ludt
Heinz Lycklama
Sheila Mallela
Roger Martin
Sunil Mehta
Pete Meier
Gary Miller
Manuel Carbajo Monje
Kevin Murphy
Yasushi Nakahara
Shigetatsu Nakao
Barry Needham
Suy Nguyen
Mary Lynne Nielsen
Patricia Oberndorf
Jim Oblinger
Peter Owens
Ed Palmer
F. G. Patterson, Jr.
Per Pedersen

Arnie Powell
Dave Pruett
Brian Purdy
Lynwood Randolph
Wendy Rauch
Brad Reed
Darryl Roberts
Mark Ruddock
Nobuo Saito
Greg Sawyer
Norman Scherer
Carl Schmiedekamp
Andy Schoka
Richard Scott
Glen Seeds
Ron Sellars
Lewis Shannon
Karen Sheaffer
Harry Singh
Pete Smith
Dukjoo Son
Vinnie Squitieri
Keith Stobie
Jong Sung Sunwoo
Sandra Swearingen
Marti Szczur
Ravi Tavakley
Eva Uristensson
Martial Van Neste
Bob Voigt
Andrew Walker
Gentry Watson
Alan Weaver
James White
John Wilber
John Williams
Arnold Winkler
Wayne Yaddow
Charles Young
George Zerdian

The following persons were members of the balloting group:

Nick Stoughton *EurOpen Institutional Representative*
Robert Boucher *Uniforum Institutional Representative*

Norman Aaronson	Allen L. Hankinson	Robert Sarr
Michelle Aden	Barry Hedquist	Andrew M. Schoka
Lynda Allen	Hans H. Heilborn	Fritz Schulz
Bengt Asker	John L. Hill	Richard L. Scott
Ralph Barker	James C.M. Ho	Peter Smith
Richard M. Bergman	Andrew R. Huber	Jeff Stevenson
Andy R. Bihain	Richard Hughes-Rowlands	Sandra Swearingen
Robert Bismuth	Jim Isaak	James G. Tanner
Keith Brophy	Petr Janecek	Ravi Tavakley
Dawn Burnett	Hal Jespersen	Donn S. Terry
George S. Carson	Derek Kaufman	Andrew T. Twigger
Stephan M. Chan	Judy Kerner	Mark-René Uchida
Kilnam Chon	Lorraine C. Kevra	Martial Van Neste
William Corwin	Martin J. Kirk	Andrew Walker
Fred D. Crouner	Greger Leijonhufvud	Stephen R. Walli
Dave Decot	Kevin Lewis	Paul Wanish
Shane Deichman	Lee W. Lucas	Bruce Weiner
Stephen L. Diamond	Roger Martin	Andrew E. Wheeler
Ron Elliott	Roland McGrath	Alex White
Richard W. Elwood	Pete Meier	John R. Williams
Philip H. Enslow	Gary W. Miller	Peter Wishart
Donna K. Fisher	John S. Morris	Charles R. Young
Donald Folland	Alok C. Nigam	Oren Yuen
Bob Gambrel	Patricia Oberndorf	John J. Zenor
Michel Glen	A. W. Powell	George R. Zerdian
Michael Gonzalez	Scott E. Preece	
Joe Gwinn	Wendy Rauch	

The final conditions for approval of this standard were met on May 2, 1995. This standard was conditionally approved by the IEEE Standards Board on March 16, 1995, with the following membership:

E. G. “Al” Kiener, *Chair*
Donald C. Loughry, *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril	Jim Isaak	Mary Lou Padgett
Clyde R. Camp	Ben C. Johnson	John W. Pope
Joseph A. Cannatelli	Sonny Kasturi	Arthur K. Reilly
Stephen L. Diamond	Lorraine C. Kevra	Gary S. Robinson
Harold E. Epstein	Ivor N. Knight	Ingo Rusch
Donald C. Fleckenstein	Joseph L. Koepfinger *	Chee Kiow Tan
Jay Forster *	D. N. “Jim” Logothetis	Leonard L. Tripp
Donald N. Heirman	L. Bruce McClung	
Richard J. Holleman	Marco W. Migliaro	

* Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal

Richard B. Engelman
Robert E. Hebner

Chester C. Taylor

Mary Lynne Nielsen
IEEE Standards Project Editor

Apple and Macintosh are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Open Software Foundation, OSF, the OSF logo, and Motif are registered trademarks of the Open Software Foundation, Inc.

OS/2 is a trademark of International Business Machines Corporation.

POSIX[®] is a registered certification mark of the Institute of Electrical and Electronics Engineers, Inc.

UNIX[®] is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open and the X device are trademarks of X/Open Company Ltd.

CLAUSE	PAGE
1. General	1
1.1 Scope	1
1.2 Normative References	1
1.3 Conformance	9
1.4 Test Methods	9
2. Terminology	9
2.1 Conventions	9
2.2 Definitions	10
3. POSIX Open System Environment (OSE)	14
3.1 POSIX OSE — General Objectives	15
3.2 POSIX OSE Reference Model	17
3.3 POSIX OSE Services	25
3.4 POSIX OSE Standards	26
3.5 POSIX Profiles	28
3.6 PIIs	28
4. POSIX OSE Services	29
4.1 Language Services	31
4.2 Core System Services	37
4.3 Communication Services	48
4.4 Database Services	61
4.5 Data Interchange Services	70
4.6 Transaction Processing Services	77
4.7 User Command Interface Services	84
4.8 Character-Based User Interface Services	90
4.9 Windowing System Services	95
4.10 Graphics Services	107
4.11 Application Software Development Support Services	118
5. POSIX OSE Cross-Category Services	122
5.1 Internationalization Services	122
5.2 System Security Services	133
5.3 Systems Management Services	138
6. Profiles	148
6.1 Scope	148
6.2 Concepts Related to Profiles	149
6.3 Guidance to Profile Developers	150
6.4 Types of Profiles	153

CLAUSE	PAGE
7. POSIX SP Profiling Efforts	154
7.1 Introduction.....	154
7.2 Multiprocessing Systems Platform Profiles	155
7.3 POSIX Interactive Systems AEP	156
7.4 Supercomputing AEP	157
7.5 Realtime AEPs	158
Annex A (Informative) Bibliography	160
Annex B (Informative) Standards Organizations and Contact Information	164

IEEE Guide to the POSIX[®] Open System Environment (OSE)

1. General

1.1 Scope

This guide provides

- 1) Definitions of the concepts of application portability, application interoperability, data portability, and user portability
- 2) Descriptions of services needed in the areas of application portability, application interoperability, data portability, and user portability
- 3) A survey of existing standards that address these objectives
- 4) Identification of those areas where formal standards do not exist and discussion of near-term strategies for filling these gaps
- 5) Guidance on assembling these standards into profiles

This guide includes no mandatory requirements.

1.2 Normative References

The following standards contain provisions that, through references in this text, constitute provisions of this guide. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on these standards are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

NOTE — All formal international, regional, and national standards cited in this guide are listed in this clause. Only those JTC 1 standards that have achieved DIS or ITU-T Recommendation status are listed. Only those national body standards that have completed the process of adoption are listed, and only when there is no international or regional standard covering the same area. All other specifications or references cited in this guide are given as bibliographic entries in Annex A.

{1}ISO/IEC 646: 1991, ¹ *Information processing— ISO 7-bit coded character set for information interchange*.

¹ISO documents can be obtained from the ISO office, 1, rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.

- {2}ISO/IEC 1539:1991, *Information technology—Programming languages—Fortran.*
- {3}ISO 1989: 1985 [ANSI X3.23-1985 (Reaff. 1991)], *Programming languages—COBOL.*
- {4}ISO 2014: 1976, *Information processing—Writing of calendar dates in allnumeric form.*²
- {5}ISO/IEC 2022: 1994, *Information technology—Character code structure and extension techniques.*
- {6}ISO 3307: 1975, *Information processing—Representation of time of the day.*²
- {7}ISO 4031: 1987, *Information processing—Representation of local time differentials.*²
- {8}ISO 4217: 1990, *Codes for the representation of currencies and funds.*
- {9}ISO 4873: 1991, *Information technology— ISO 8-bit code for information interchange—Structure and rules for implementation.*
- {10}ISO 6093: 1985, *Information processing—Representation of numerical values in character strings for information interchange.*
- {11}ISO 6160: 1979 [ANSI X3.53-1976 (Reaff. 1993)], *Programming languages—PL/I.*
- {12}ISO/IEC 6429: 1992, *Information technology—Control functions for coded character sets.*
- {13}ISO/IEC 6522: 1992 [ANSI X3.74-1987 (Reaff. 1993)], *Programming languages—Information technology—PL/I general purpose subset.*
- {14}ISO 6936: 1988, *Information processing—Conversion between the two coded character sets of ISO 646 and 6937-2 and the CCITT international telegraph alphabet No. 2 (ITA2).*
- {15}ISO/IEC 6937: 1994, *Information technology—Coded graphic character set for text communication—Latin alphabet.*
- {16}ISO/IEC 7185: 1990, *Information technology—Programming languages—Pascal.*
- {17}ISO/IEC 7350: 1991, *Information technology—Registration of repertoires of graphic characters from ISO 10367.*
- {18}ISO/IEC 7498: 1994, *Information processing systems—Open Systems Interconnection—Basic Reference Model: The Basic Model.*
- {19}ISO 7498-2: 1989, *Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 2: Security Architecture.*
- {20}ISO 7942: 1985, *Information processing systems—Computer graphics—Graphical Kernel System (GKS) functional description.*
- {21}ISO/IEC 8072: 1994, *Information processing systems—Open Systems Interconnection—Transport service definition.*
- {22}ISO/IEC 8073: 1992, *Information technology—Telecommunications and information exchange between systems—Open Systems Interconnection—Protocol for providing the connection-mode transport service.*

²Please note that this standard is currently withdrawn.

{23}ISO 8327: 1987, *Information processing systems—Open Systems Interconnection—Basic connection oriented session protocol specification.*

{24}ISO 8485: 1989, *Programming languages—APL.*

{25}ISO 8571, *Information processing systems—Open Systems Interconnection—File Transfer, Access and Management.*

{26}ISO 8601:1988, *Data elements and interchange formats—Information interchange—Representation of dates and times.*

{27}ISO 8602:1987, *Information processing systems—Open Systems Interconnection—Protocol for providing the connectionless-mode transport service.*

{28}ISO 8613, *Information processing—Text and office systems—Office Document Architecture (ODA) and interchange format.*

{29}ISO/IEC 8632: 1992, *Information technology—Computer graphics—Metafile for the storage and transfer of picture description information.*

{30}ISO 8649:1988, *Information processing systems—Open Systems Interconnection—Service definition for the Association Control Service Element.*

{31}ISO 8650: 1988, *Information processing systems—Open Systems Interconnection—Protocol specification for the Association Control Service Element.*

{32}ISO 8651, *Information processing systems—Computer graphics—Graphical Kernel System (GKS) language bindings.*

{33}ISO 8652: 1987 (ANSI/MIL 1815A-1983), *Programming languages—Ada.*

{34}ISO/IEC 8802-3:1993 (802.3, 1993 Edition), *Information technology—Local and metropolitan area networks—Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.*

{35}ISO/IEC 8802-4: 1990 (802.4-1990), *Information processing systems—Local area networks—Part 4: Token-passing bus access method and physical layer specifications.*

{36}ISO/IEC 8802-5: 1992 (802.5-1992), *Information processing systems—Local and metropolitan area networks—Part 5: Token ring access method and physical layer specifications.*

{37}ISO 8805: 1988, *Information processing systems—Computer graphics—Graphical Kernel System for Three Dimensions (GKS-3D) functional description.*

{38}ISO/IEC 8806-1: ..., ³ *Information processing systems—Computer graphics—Graphical Kernel System for Three Dimensions (GKS-3D) language bindings—Part 1: FORTRAN.*

{39}ISO/IEC 8806-4: 1991, *Information technology—Computer graphics—Graphical Kernel System for Three Dimensions (GKS-3D) language bindings—Part 4: C.*

{40}ISO/IEC 8823-1: 1994, *Information technology—Open Systems Interconnection—Connection-oriented presentation protocol: Protocol specification.*

³Presently at the state of Draft International Standard.

- {41}ISO/IEC 8824: 1990, *Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1)*.
- {42}ISO/IEC 8825: 1990, *Information technology—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.
- {43}ISO 8859-1: 1987, *Information processing—8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1*.
- {44}ISO 8879: 1986, *Information processing—Text and office systems—Standard Generalized Markup Language (SGML)*.
- {45}ISO 9040: 1990, *Information technology—Open Systems Interconnection—Virtual Terminal Basic Class Service*.
- {46}ISO 9041, *Information technology—Open Systems Interconnection—Virtual Terminal Basic Class Protocol*.
- {47}ISO 9069:1988, *Information processing—GML support facilities—SGML Document Interchange Format (SDIF)*.
- {48}ISO/IEC 9075:1992 (ANSI X3.135-1992), *Information technology—Database languages—SQL*.
- {49}ISO 9241, *Ergonomic requirements for office work with visual display terminals (VDTs)*.
- {50}ISO/IEC 9314, *Information technology—Fibre Distributed Data Interface (FDDI)*.
- {51}ISO 9316:1989, *Information processing systems—Small Computer System Interface (SCSI)*.
- {52}ISO/IEC 9541, *Information technology—Font information interchange*.
- {53}ISO/IEC 9548:, ...⁴ *Information processing systems—Open Systems Interconnection—Session connectionless protocol to provide the connectionless-mode session service*.
- {54}ISO/IEC 9576:1991, *Information technology—Open Systems Interconnection—Connectionless presentation protocol specification*.
- {55}ISO/IEC 9579, *Information technology—Open Systems Interconnection—Remote Database Access*.
- {56}ISO/IEC 9592, *Information processing systems—Computer graphics—Programmer's Hierarchical Interactive Graphics System (PHIGS)*.
- {57}ISO/IEC 9593, *Information processing systems—Computer graphics—Programmer's Hierarchical Interactive Graphics System (PHIGS) language bindings*.
- {58}ISO/IEC 9594-1:1990, *Information technology—Open Systems Interconnection—The Directory—Part 1: Overview of concepts, models and services*.
- {59}ISO/IEC 9594-8:1990, *Information technology—Open Systems Interconnection—The Directory—Part 8: Authentication framework*.
- {60}ISO/IEC 9596-1:1991, *Information technology—Open Systems Interconnection—Common management information protocol—Part 1: Specification*.

⁴Presently at the state of Draft International Standard.

{61}ISO/IEC 9636:1991, *Information technology—Computer graphics—Interfacing techniques for dialogues with graphical devices (CGI)—Functional specification.*

{62}ISO/IEC 9637-2:1992, *Information technology—Computer graphics—Interfacing techniques for dialogues with graphical devices (CGI)—Data stream binding—Part 2: Binary encoding.*

{63}ISO/IEC 9638-3:1994, *Information technology—Computer Graphics—Interfacing techniques for dialogues with graphical devices (CGI)—Language bindings—Part 3: Ada.*

{64}ISO 9735:1988, *Electronic data interchange for administration, commerce and transport (EDIFACT)—Application level syntax rules (Amended and reprinted 1990).*

{65}ISO/IEC 9804:1994, *Information technology—Open Systems Interconnection—Service definition for the commitment, concurrency and recovery service element.*

{66}ISO/IEC 9805-1:1994, *Information technology—Open Systems Interconnection—Protocol for the Commitment, Concurrency and Recovery service element—Protocol Specification.*

{67}ISO/IEC 9899: 1990, *Programming languages—C.*

{68}ISO/IEC 9945-1:1990 (1003.1-1990), *Information technology—Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API) [C Language].*

{69}ISO/IEC 9945-2:1993 (1003.2-1992), *Information technology—Portable Operating System Interface (POSIX®)—Part 2: Shell and Utilities.*

{70}ISO/IEC 9995:1994, *Information technology—Keyboard layouts for text and office systems.*

{71}ISO/IEC 10021, *Information technology—Text Communication—Message-Oriented Text Interchange Systems (MOTIS).*

{72}ISO/IEC 10026-1:1992, *Information technology—Open Systems Interconnection—Distributed Transaction Processing—Part 1: OSI TP Model.*

{73}ISO/IEC 10027:1990, *Information technology—Information Resource Dictionary System (IRDS) framework.*

{74}ISO/IEC 10164-4:1992 [ITU-T X.733 (1992)], *Information technology—Open Systems Interconnection—Systems management: Alarm reporting function.*

{75}ISO/IEC 10164-5:1993 [ITU-T X.734 (1992)], *Information technology—Open Systems Interconnection—Systems management: Event Report Management Function.*

{76}ISO/IEC 10164-6:1993 [ITU-T X.735 (1992)], *Information technology—Open Systems Interconnection—Systems Management: Log control function.*

{77}ISO/IEC 10164-7:1992 [ITU-T X.736 (1992)], *Information technology—Open Systems Interconnection—Systems Management: Security alarm reporting function.*

{78}ISO/IEC 10164-10:...,⁵ [ITU-T X.742 (1993)], *Information processing systems—Open Systems Interconnection—Systems Management: Usage Metering Function.*

⁵Presently at the state of Draft International Standard.

{79}ISO/IEC 10179:..., ⁶ *Information technology—Text and office systems—Document Style Semantics and Specification Language (DSSSL)*.

{80}ISO/IEC 10180, : ..., ⁶ *Information technology—Text communication—Standard Page Description Language (SPDL)*.

{81}ISO/IEC 10206: 1991, *Information technology—Programming languages—Extended Pascal*.

{82}ISO/IEC 10279: 1991, *Information technology—Programming languages—Full BASIC*.

{83}ISO/IEC 10303, *Industrial automation systems and integration—Product data representation and exchange*.

{84}ISO/IEC 10367: 1991, *Information technology—Standardized coded graphic character sets for use in 8-bit codes*.

{85}ISO/IEC 10514, : ..., ⁶ *Information technology—Programming languages—Modula-2*.

{86}ISO/IEC ISP 10607, *Information technology—International Standardized Profiles AFTnn—File Transfer, Access and Management*.

{87}ISO/IEC 10641:1993, *Information technology—Computer graphics and image processing—Conformance testing of implementations of graphics standards*.

{88}ISO/IEC 10646-1:1993, *Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane*.

{89}ISO/IEC 10728:1993, *Information technology—Information Resource Dictionary System (IRDS) Services Interface*.

{90}ISO/IEC 10744: 1992, *Information technology—Hypermedia/Time-based Structuring Language (HyTime)*.

{91}ISO/IEC 11072: 1992, *Information technology—Computer graphics—Computer Graphics Reference Model*.

{92}ISO/IEC 11730:1994, *Information technology—Programming languages—Form Interface Management System (FIMS)*.

{93}ISO/IEC 12087, *Information technology—Computer graphics and image processing—Image Processing and Interchange (IPI)—Functional specification*.

{94}ISO/IEC 12088-4: ..., ⁶ *Information technology—Computer graphics and image processing—Image processing and interchange (IPI)—Application program interface language bindings—Part 4: C*.

{95}ISO/IEC 12089: ..., ⁶ *Information technology—Computer graphics and image processing—Encoding for the Standard Image Processing and Interchange (IPI)—Encoding for the Image Interchange Facility (IIF)*.

{96}ISO/IEC 12227:1995, *Information technology—Programming languages—SQL/Ada Module Description Language (SAMeDL)*.

{97}ISO/IEC 13210:1994 (IEEE Std 1003.3-1991), *Information technology—Test methods for measuring conformance to POSIX[®]*.

{98}ISO/IEC 13211-1: 1995, *Information technology—Programming languages—Prolog—Part 1: General core*.

⁶Presently at the state of Draft International Standard.

{99}ITU-T Recommendation 1.120 (1993),⁷ *Integrated services digital networks (ISDNs)*.

{100}ITU-T Recommendation T.61 (1993), *Character repertoire and coded character sets for the international teletex service*.⁸

{101}ITU-T Recommendation X.25 (1993), *Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*.

{102}ITU-T Recommendation X.400 (1993), *Message handling services: Message handling system and service overview*.

{103}ANSI X3.9-1978 (Reaff. 1989),⁹ *American National Standard Programming Language FORTRAN*.

{104}ANSI X3.62-1987 (Reaff. 1993), *Information Systems—Optical Character Recognition (OCR)—Paper Used in OCR Systems*.

{105}ANSI X3.102-1992, *Information Systems—Data Communication Systems and Services—User-Oriented Performance Parameters*.

{106}ANSI X3.138-1988, *Information Systems—Information Resource Dictionary System (IRDS)*.

{107}ANSI X3.131-1994, *Information Systems—Small Computer Systems Interface-2 (SCSI-2)*.

{108}ANSI X3.141-1987 (Reaff. 1992), *Information Systems—Data Communication Systems and Services—Measurement Methods for User-Oriented Performance Evaluation*.

{109}ANSI X3.168-1989, *Information Systems—Database Language—Embedded SQL*.

{110}ANSI X3.185-1992, *Information Systems—Information Resource Dictionary Systems—IRDS Services Interface*.

{111}ANSI X3.195-1991, *Information Systems—Information Resource Dictionary Systems (IRDS)—Export/Import File Format*.

{112}ANSI/ASME Y14.26M-1989, *Digital Representation for Communication of Product Definition Data*.

{113}ECMA TR-47,¹⁰ *Configuration Management Service Definition*.

{114}ECMA 138, *Security in Open Systems—Data Elements and Service Definitions*.

{115}GB 2312-1980, *China State Bureau of Standards*.¹¹ *Coded Chinese Graphic Character Set for Information Interchange*.

⁷ITU-T documents can be obtained from the ITU-T General Secretariat, International Telecommunications Union, Sales Section, Place des Nations, CH-1211, Genève 20, Switzerland/Suisse.

⁸Please note that this recommendation is currently withdrawn.

⁹ANSI documents can be obtained from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.

¹⁰ECMA documents can be obtained from ECMA, 114 rue du Rhône, CH-1204 Genève, Switzerland/Suisse.

¹¹CSBS documents can be obtained from the China State Bureau of Standards, P.O. Box 8010, 42 Hi Chun Road, Haidian District, Beijing 100088, China.

{116}IEEE 1003.1b-1993, ¹² *IEEE Standard, or Information Technology-Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API) [C Language]—Amendment 1: Realtime Extensions.*

NOTE: This standard was formerly called draft P1003.4.

{117}IEEE Std 1003.1c-1995, *IEEE Standard for Information Technology-Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API)—Amendment 2: Threads Extension IC Language].*

NOTE: This standard was formerly called draft P1003.4a.

{118}IEEE Std 1003.2d-1994, *IEEE Standard for Information Technology-Portable Operating System Interface (POSIX®)—Part 2: Shell and Utilities—Amendment 1: Batch Environment.*

NOTE: This standard was formerly called draft P1003.15.

{119}IEEE Std 1003.5-1992, *IEEE Standard for Information Technology—POSIX® Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API).*

{120}IEEE Std 1003.9-1992, *IEEE Standard for Information Technology—POSIX® FORTRAN 77 Language Interfaces—Part 1: Binding for System Application Program Interface (API).*

{121}IEEE Std 1003.10-1995, *IEEE Standard for Information Technology-POSIX® Supercomputing Application Environment Profile.*

{122}IEEE Std 1076-1993, *IEEE Standard VHDL Language Reference Manual.*

{123}IEEE Std 1224-1993, *IEEE Standard for Information Technology—Open Systems Interconnection (OSI) Abstract Data Manipulation—Application Program Interface (API) [Language Independent].*

{124}IEEE Std 1224.1-1993, *IEEE Standard for Information Technology—X.400-Based Electronic Messaging—Application Program Interface (API) [Language Independent].*

{125}IEEE Std 1224.2-1993, *IEEE Standard for Information Technology—Directory Services—Application Program Interface (API) [Language Independent].*

{126}IEEE Std 1238.1-1995, *IEEE Standard for Information Technology—File Transfer, Access, and Management Services—Application Program Interface (API) [C Language Binding].*

{127}IEEE Std 1295-1993, *IEEE Standard for Information Technology—X Window System—Modular Toolkit Environment.*

{128}IEEE Std 1327-1993, *IEEE Standard for Information Technology—Open Systems Interconnection (OSI) Abstract Data Manipulation C Language Interfaces—Binding for Application Program Interface (API).*

{129}IEEE Std 1327.1-1993, *IEEE Standard for Information Technology—X.400-Based Electronic Messaging C Language Interfaces—Binding for Application Program Interface (API).*

{130}IEEE Std 1327.2-1993, *IEEE Standard for Information Technology—Directory Services C Language Interfaces—Binding for Application Program Interface (API).*

¹²This standard is available from the IEEE 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. Telephone: 1 (800) 678-IEEE or +1 (908) 981-1393 (outside USA).

{131} IEEE Std 1387.2-1995, *IEEE Standard for Information Technology-Portable Operating System Interface (POSIX®) System Administration-Part 2: Software*.

{132} JIS X0208-1990, Japanese Industrial Standard. ¹³ *Code of the Japanese graphic character set for information interchange*.

{133} JIS X0212-1990, Japanese Industrial Standard. *Code of the supplementary Japanese graphic character set for information interchange*.

{134} KS C 5601-1987, Korean Bureau of Standards. ¹⁴ *Korean Graphic Character Set for Information Interchange*.

1.3 Conformance

It is not appropriate to claim conformance to this guide because it contains no mandatory requirements. This guide is intended to be used only as a source of reference material.

1.4 Test Methods

Not applicable.

2. Terminology

2.1 Conventions

2.1.1 Editorial Conventions

This guide uses the following editorial and typographical conventions. A summary of typographical conventions is shown in Table 2.1.

Table 2.1—Typographical Conventions

Reference	Example
C-Language function	<i>system ()</i>
Cross-Reference: Annex	Annex A
Cross-Reference: Clause	2.3
Cross-Reference: Other standard	ISO/IEC 9999-1 { <i>n</i> }
Cross-Reference: Section	Section 2
Cross-Reference: Subclause	2.3.4, 2.3.4.5, 2.3.4.5.6
Defined term	(see text)
Figure reference	Figure 7-1
Table reference	Table 6-1
Utility name	awk

¹³JIS documents can be obtained from the Japanese Standards Association, 1-24, Akasaka 4-chome, Minato-Ku, Tokyo, Japan 107.

¹⁴KBS documents can be obtained from the Korean Bureau of Standards, 2 Chung-ang-dong Kwach'on-city, Kyonggi-do 171-11, Republic of Korea.

Numbers within braces, such as “ISO/IEC 9945-1 :1990 {68},” represent cross-references to the normative references clause (see 1.2). If the number is preceded by a B, it represents a bibliographic entry (see Annex A). See 2.1.2 for a further description of the references to POSIX working groups and standards.

Defined terms are shown in three styles, depending on context:

- 1) Terms defined in 2.2.1 and 2.2.2 are expressed as subclause titles. Alternative forms of the terms appear in [brackets].
- 2) The initial appearances of other terms, applying to a limited portion of the text, are in *italics*.
- 3) Subsequent appearances of the term are in the Roman font.

2.1.2 POSIX

The term “POSIX” has been evolving into a term with a number of different meanings. This subclause attempts to define the word and some related terms. The intent is to ensure that the term POSIX is used in a useful and predictable manner in this guide.

As background, note that POSIX is sometimes used to denote the formal standard ISO/IEC 9945-1 :1990 {68}, sometimes to denote that standard plus related standards and drafts emerging from IEEE PASC working groups (such as P1003, P1201, P1224, P1278, P1387, etc.), and sometimes to denote the groups themselves. In all those cases, POSIX is used as a noun.

This guide uses the term “POSIX” only as an adjective, and uses it only in well-defined ways. This subclause serves as a preview of the usages in this guide of POSIX terms. (These terms are defined, formally or informally, in subsequent clauses, and the reader will be referred to those clauses as appropriate.)

This guide refers to the original POSIX standard by its standard designation, ISO/IEC 9945-1 :1990 {68}, and not by the term POSIX.

The IEEE groups developing standards related to IEEE P1003 are called, in this guide, *IEEE P1003 .n working groups*. Examples are the IEEE working groups P1003.2, P1003.3, etc. The names of the groups are sometimes abbreviated POSIX.2, POSIX.3, etc., but this convention is not used by this guide; confusion could result when the IEEE P1003 decimal number does not match the ISO/IEC 9945 part number. Furthermore, there are other IEEE working groups, such as P1224, that do not use the POSIX prefix. Therefore, all IEEE projects and working groups are referred to uniformly as IEEE *Pnnnn*.

The standards emerging out of the POSIX working groups are referred to by their formal names (e.g., IEEE Std 1003.2-1992 or IEEE P1003.10/D9) and are called either *POSIX base standards* or *POSIX SPs*.

2.2 Definitions

2.2.1 Terminology

For the purposes of this guide, the following definitions apply:

2.2.1.1 implementation defined: An indication that the implementation shall define and document the requirements for correct program constructs and correct data of a value or behavior.

2.2.1.2 informative: Providing or disclosing information; instructive

Used in standards to indicate a portion of the text that poses no requirements; the opposite of *normative*.

2.2.1.3 may: An indication of an optional feature

With respect to implementations, the word may is to be interpreted as an optional feature that is not required in this guide, but can be provided.

2.2.1.4 normative: Of, pertaining to, or prescribing a norm or a standard.

Used in standards to indicate a portion of the text that poses requirements.

2.2.1.5 should: With respect to implementations, an indication of an implementation recommendation, but not a requirement.

2.2.2 General Terms

For the purposes of this guide, the following definitions apply.

2.2.2.1 accredited standards development organization: An organization recognized as a standards development organization by ISO, IEC, ITU-T, or recognized as a standards development organization by one of the member bodies of one of these three organizations.

2.2.2.2 application: The use of capabilities provided by an information system specific to the satisfaction of a set of user requirements.

NOTE — These capabilities include hardware, software, and data.

2.2.2.3 application environment profile (AEP): A profile specifying a complete and coherent specification of the Open System Environment (OSE), in which the standards, options, and parameters chosen are necessary to support a class of applications.

2.2.2.4 application platform: A set of resources, including hardware and software, that support the services on which application software will run.

The application platform provides services at its interfaces that, as much as possible, make the specific characteristics of the platform transparent to the application software.

2.2.2.5 application program interface (API): The interface between the application software and the application platform, across which all services are provided.

2.2.2.6 application software: Software that is specific to an application and is composed of programs, data, and documentation.

2.2.2.7 base standard: An approved international standard, technical report, ITU-T Recommendation, or national standard.

2.2.2.8 communication interface: That part of the API devoted to communications with other application software, external data transport facilities, and devices.

2.2.2.9 communication services interface (CSI): The boundary across which access to services for interaction between internal application software entities and application platform external entities is provided.

2.2.2.10 component profile: A profile that is made up of a formally defined subset of a single standard.

2.2.2.11 cross-category services: A set of tools or features or both that has a direct effect on the operation of one or more components of the OSE, but is not in and of itself a stand-alone component.

2.2.2.12 emerging standard: A specification that is under consideration by an accredited standards development organization, but that has not completed the process of approval by the sponsoring body.

Emerging standards are often subject to significant change prior to approval.

2.2.2.13 external environment: A set of entities external to the application platform with which services are provided.

External entities include people, exchangeable media that is not mounted in the platform, communication wiring, and other platforms.

2.2.2.14 external environment interface (EEI): The interface between the application platform and the external environment across which services are provided.

The EEI is defined primarily in support of system and application interoperability.

The primary services present at the EEI comprise

- Human/Computer interaction services
- Information services
- Communication services

2.2.2.15 hardware: Physical equipment used in data processing, as opposed to programs, procedures, rules, and associated documentation.

2.2.2.16 harmonization: The process of ensuring that profiles do not overlap or conflict.

2.2.2.17 human/computer interface (HCI): The boundary across which physical interaction between a human being and the application platform takes place.

2.2.2.18 information services interface (ISI): The boundary across which external, persistent storage service is provided.

2.2.2.19 interface : A shared boundary between two functional entities.

A standard specifies the services in terms of the functional characteristics and behavior observed at the interface. The standard is a contract in the sense that it documents a mutual obligation between the service user and provider and assures a stable definition of that obligation.

2.2.2.20 internationalization : The process of designing and developing an implementation with a set of features, functions, and options intended to satisfy a variety of cultural environments.

(See also *localization* in 2.2.2.26.)

2.2.2.21 interoperability : The ability of two or more systems to exchange information and to use the information that has been exchanged mutually.

2.2.2.22 language-binding API specification : A specification that documents the source code method, consistent with a specific programming language, used by an application to access services provided by an application platform.

2.2.2.23 language-independent service specification: A specification that defines a set of required functional semantics independent of the syntax and semantics of a programming language.

2.2.2.24 local adaptation : The process of modifying a product that is specific to one culture to make it specific to another culture.

2.2.2.25 locale : The definition of the user environment that depends on language and cultural conventions.

2.2.2.26 localization : The process of utilizing internationalization features to adapt an internationalized product to a specific cultural environment.

(See also *internationalization* in 2.2.2.20.)

2.2.2.27 open specifications: Specifications that are maintained by an organization that uses an open, public consensus process to accommodate new technologies and user requirements over time.

2.2.2.28 open system: A system that implements sufficient open specifications or standards for interfaces, services, and supporting formats to enable properly engineered application software

- To be ported with minimal changes across a wide range of systems from one or more suppliers
- To interoperate with other applications on local and remote systems
- To interact with people in a style that facilitates user portability

2.2.2.29 open system API: A combination of standards-based interfaces specifying a complete interface between an application program and the underlying application platform.

2.2.2.30 open system environment (OSE): A comprehensive set of interfaces, services, and supporting formats, plus user aspects for interoperability or for portability of applications, data, or people, as specified by information technology standards and profiles.

2.2.2.31 operating system software: Application-independent software that supports the running of application software and manages the resources of the application platform.

2.2.2.32 performance: A measure of a computer system or subsystem to perform its functions; for example, response time, throughput, or number of transactions per second.

The efficiency of a system in accomplishing pieces of work is an attribute of performance.

2.2.2.33 performance requirement: A requirement that specifies a performance characteristic that a system or system component must possess; for example, speed, accuracy, frequency.

2.2.2.34 platform internal interface (PH): The interface between application platform service components within that platform.

2.2.2.35 platform profile: A profile whose focus is on functionality and interfaces for a particular type of platform, which may be a single processor shared by a group of applications or a large distributed system with each application dedicated to a single processor.

2.2.2.36 portability (application software): The ease with which application software and data can be transferred from one application platform to another.

2.2.2.37 POSIX Standardized Profile (POSIX SP): A standardized profile that specifies the application of certain POSIX base standards in support of a class of applications and does not require any departure from the structure defined by the reference model for POSIX systems in this guide.

2.2.2.38 process: An address space and one or more threads of control that execute within that address space, and their required system resources.

2.2.2.39 profile: A set of one or more base standards and, where applicable, the identification of chosen classes, subsets, options, and parameters of those base standards that are necessary for accomplishing a particular function.

2.2.2.40 programming language API specification: The interface between applications and application platforms traditionally associated with programming language specifications, such as program control, math functions, string manipulation, etc.

2.2.2.41 protocol: A set of semantic and syntactic rules that determine the behavior of entities that interact.

2.2.2.42 public specifications: Specifications that are available, without restriction, to anyone for implementation, sublicensing, and distribution (i.e., sale) of that implementation.

2.2.2.43 reference model: A structured collection of concepts and their relationships that scope a subject and enable the partitioning of the relationships into topics relevant to the overall subject and that can be expressed by a common means of description.

2.2.2.44 scalability: The ability to provide functionality up and down a graduated series of application platforms that differ in speed and capacity.

2.2.2.45 security: The protection of computer resources (e.g., hardware, software, and data) from accidental or malicious access, use, modification, destruction, or disclosure.

Tools for the maintenance of security are focused on availability, authentication, accountability, confidentiality, and integrity.

2.2.2.46 service: A distinct part of the functionality that is provided by an entity on one side of an interface to an entity on the other side of the interface.

2.2.2.47 software: The programs, procedures, rules, and any associated documentation pertaining to the operation of an information processing system.

2.2.2.48 specification: A document that prescribes, in a complete, precise, verifiable manner, the requirements, design, behavior, or characteristics of a system or system component.

2.2.2.49 standard: A document, established by consensus and approved by an accredited standards development organization, that provides, for common and repeated use, rules, guidelines, or characteristics for activities or their results, aimed at the achievement of the optimum degree of order and consistency in a given context.

2.2.2.50 standardized profile: A balloted, formal, harmonized document that specifies a profile.

2.2.2.51 standard development organization: An accredited organization that formally develops and coordinates standards for use by a community.

2.2.2.52 thread: A single flow of control within a process.

2.2.2.53 transaction: A unit of work consisting of an arbitrary number of individual operations, all of which will either complete successfully or abort with no effect on the intended resources.

A transaction has well-defined boundaries. A transaction starts with a request from the application program and either completes successfully (commits) or has no effect (abort). Both the commit and abort signify completion of a transaction.

2.2.2.54 validation: The process of testing an application or system to ensure that it conforms to its specification.

2.2.3 Abbreviations

For the purposes of this guide, the following abbreviations apply:

2.2.3.1 AEP: Application Environment Profile.

2.2.3.2 API: : Application Program Interface.

2.2.3.3 ASC: Accredited Standards Committee.

2.2.3.4 CSI: Communications Service Interface.

2.2.3.5 EEI: External Environment Interface.

2.2.3.6 HCI: Human/Computer Interface.

2.2.3.7 ISI: Information Service Interface.

2.2.3.8 ISP: International Standardized Profile.

2.2.3.9 OSE: Open System Environment.

2.2.3.10 OSI: Open System Interconnect.

2.2.3.11 PASC: The Portable Applications Standards Committee of the IEEE Computer Society.

2.2.3.12 PII: Platform Internal Interface.

2.2.3.13 SP: Standardized Profile.

3. POSIX Open System Environment (OSE)

The POSIX OSE provides a context for user services and standards specification. It provides a minimum standard set of conceptual information-system building blocks with associated interfaces and functionality. The POSIX OSE consists of a reference model, service definitions, standards, and profiles.

These POSIX OSE concepts are intended to be conventional within computer science. The intention is not to break new ground, but to establish a minimum and unambiguous terminology and a set of concepts for identification and resolution of portability and interoperability issues.

The POSIX OSE is defined in five parts:

- 1) General objectives that apply to the POSIX OSE as a whole are identified in 3.1.
- 2) A reference model is developed that unambiguously identifies the system under consideration for purposes of specification. The POSIX OSE reference model described in 3.2 defines system elements and key interfaces between them that are related to application software portability and interoperability.
- 3) Using the interfaces identified in the reference model, each clause of Section 4 categorizes and describes the basic services available to users across each interface. The services are defined in a generic way, based on the reference model, user requirements, and current industry practice, rather than any given implementation. Each clause of Section 4 begins with a more detailed and specialized version of the reference model to provide a context for service specification. After defining the interfaces and services, each clause of Section 4 concludes with a discussion of standards that are related to the services. Definition of the services is not constrained by the availability of standards. Services that are not currently satisfied by standards are discussed in the Additional Specifications and Unaddressed Services subclauses.
- 4) Section 5 discusses issues and services that directly affect all of the service categories, such as internationalization, security, and administration.
- 5) Section 6 provides guidelines for creating profiles that address various application domains. This is a brief description of how the reference model and services are applied to a variety of existing types of systems. Section 7 describes current POSIX profiles and profiling activities.

3.1 POSIX OSE — General Objectives

The POSIX OSE described in this guide should provide services to satisfy the following objectives. These objectives may be achieved to a greater or lesser degree for any given usage.

3.1.1 Application Software Portability at the Source-Code Level

The POSIX OSE should enable application software portability at the source-code level.

Rationale: Comprehensive and consistent source-code level service specifications allow porting of applications among application platform implementations (ideally without modification). Binary portability requires too tight a coupling with the application platform implementation.

Although standards-based systems improve application portability, the standards do not guarantee that an application will be portable. Applications still have to be properly engineered to ensure maximum application portability. This allows an organization to protect its investment in existing software.

Application portability is often associated with porting an entire application at one time. *Software reuse* is a term used to describe porting only a subset of a working program into a new application. The new application may or may not be executed on the same application platform. Software reuse is an important element in achieving the benefits of application portability.

3.1.2 Data Portability

The POSIX OSE should enable transfer of stored data among application platforms.

Rationale: The ability to move stored data from one application platform to another is fundamental to achieving the objective of application portability. Since application software is defined to include both the source code and

application related data (see 2.2.2.6 and 3.2.1), porting of the application requires both the application source code and the data to be moved to the destination platform.

Data portability also directly supports the application software and application platform interoperability objective. Transfer of stored data is an important method for application software entities to exchange and make mutual use of data.

3.1.3 Application Software Interoperability and Application Platform Interoperability

The POSIX OSE should enable application software and application platform interoperability.

Rationale: Communication services and format specifications allow two entities participating in a distributed system to exchange and make mutual use of data, including

- Homogeneous systems
- Heterogeneous systems (i.e., a wide variety of hardware/software platforms)
- POSIX OSE-based and non-POSIX OSE-based systems

3.1.4 User Portability

The POSIX OSE should enable human users to operate on a wide range of application platform implementations without retraining.

Rationale: Standard methods and services for supporting human/computer interaction are a key aspect of the definition of an open system (see 2.2.2.28). Elimination of gratuitous differences in the interface that the application platform presents to the user via standards is a significant aspect of this task.

3.1.5 Accommodation of Standards

The POSIX OSE should accommodate existing, imminent, and new standards that provide interfaces, services, and formats for open systems.

Rationale: In order to accommodate standards, the POSIX OSE has to define a clear set of interfaces, upon which portability and interoperability are defined. The definition of these interfaces needs to be driven by user requirements and be as independent of the implementation technology as possible.

If the POSIX OSE were constrained to current standards and technology, it would quickly become obsolete. It would also not be capable of providing a complete set of applicable standards and profiles, as efforts to date have not yet provided a full suite of applicable standards. The POSIX OSE will evolve as standards emerge and technology changes.

3.1.6 Accommodation of New Technology

The POSIX OSE should accommodate new information system technology.

Rationale: POSIX OSE standards specify interfaces, rather than implementations. However, where the interface needs to refer to specific technology, an element of judgment is required, due to the tension between the conflicting needs for stable standards and provision for technology enhancement. The POSIX OSE should be sufficiently general to allow for technology growth and variety, and yet specific enough to act as a guide for standards development.

3.1.7 Application Platform Scalability

The POSIX OSE should be scalable to allow inclusion of platforms of varying speed, throughput, and implementation architecture (e.g., multiprocessing).

Rationale: This reflects the realities faced by the potential users of the POSIX OSE. This objective affects individual standards as well as the conditions under which various standards can or should be combined into profiles. For example, where similar services are provided by both workstation-type and supercomputer-type application platforms, the same standards should be applied to each if possible. This would enable a greater degree of portability across these specialized implementations of the application platform

3.1.8 Distributed System Scalability

The POSIX OSE should provide for distributed system scalability.

Rationale: The number of distributed system components connected should not be limited by any structural aspects of the POSIX OSE. For example, in the area of network services, the OSE standards should be such that it is possible to construct profiles (and therefore systems) in which remote and local operation and utilization of information system resources are indistinguishable, with the exception of unavoidable message transit delay. In other words, it should be possible for applications to be unaware of whether the application platform on which they are executing is local or distributed and that lack of awareness should not affect their proper operation.

3.1.9 Implementation Transparency

The POSIX OSE should provide consistent and standard interfaces to the application regardless of the underlying implementation technology.

Rationale: The mechanism for implementation of services is not visible to the service user; i.e., only the service is visible to the service user. The POSIX OSE calls for interfaces that are specified in such a way as to be independent of the implementation technology. The interface specifications are the complete definition of the application platform. The specifications should be complete enough that one does not need to see the source code to program.

3.1.10 User Functional Requirements

The POSIX OSE should reflect the full scope of the functional requirements of the user, within the context of the preceding objectives.

Rationale: The POSIX OSE will provide the context within which application software portability and interoperability requirements objectives are satisfied. The user of the information system is the final authority with respect to completeness of the services required to achieve full portability. In particular, the range of services required is broader than those traditionally identified as operating system services.

3.2 POSIX OSE Reference Model

The POSIX OSE is based on a reference model with the full (potentially distributed) information system as its scope. As such, it spans the gap between requirement specification and the design of any specific system that uses information technology. The reference model provides a set of conventions and concepts, mutually agreed upon between the information system user and provider communities. This common understanding is key to achieving application software portability and system interoperability, and it may encourage software reuse. It may lead to more compact and correct procurement specifications.

An information system reference model addresses conflicting requirements similar to those encountered in traditional architectural disciplines. The reference model has to be structured enough to encourage the generation and use of standards and standard components, yet it also has to be flexible enough to accommodate tailored and special purpose components necessary to meet real user requirements.

The POSIX OSE reference model is a set of concepts, interfaces, entities, and diagrams that provides a basis for specification of standards. It provides guidance and direction for future standardization and integration efforts. In order

for the POSIX OSE to evolve and mature, it is necessary for the reference model to provide insights into those services and capabilities for which standards do not currently exist and for which appropriate standardization activities cannot be identified.

The POSIX OSE reference model is described from the user perspective; i.e., the reference model records the perception of the application platform user (mental model) of the overall large distributed system used to support the user enterprise. This point of view assures that

- Information technology users will have the proper services to meet their requirements
- Information technology vendor implementations will not be constrained unnecessarily

Figure 3.1 depicts the POSIX OSE reference model. The model identifies three entities (Application Software, Application Platform, and External Environment) and two interfaces between them, identified as the API and the EEI. The application platform provides API and EEI services across the associated interfaces.

This model has been generalized to such a degree that it can accommodate a wide variety of general- and special-purpose systems. A more detailed discussion of each service category is provided in Section 4. The service specification has been defined to be flexible enough to allow subsets or extensions for each category as needed. As a result, the POSIX OSE reference model is able to accommodate a variety of architectures and standardization approaches. It should be possible to show where any relevant standard fits within the reference model.

For the purposes of this guide, the standards discussed address only interfaces between entities, including services and supporting formats offered across those interfaces. The interface specification defines a convention adopted to represent the function offered across the interface in both directions. Note that no set of standards can, by itself, assure portability of specific applications. Applications need to be properly engineered with an explicit portability objective in order to achieve it.

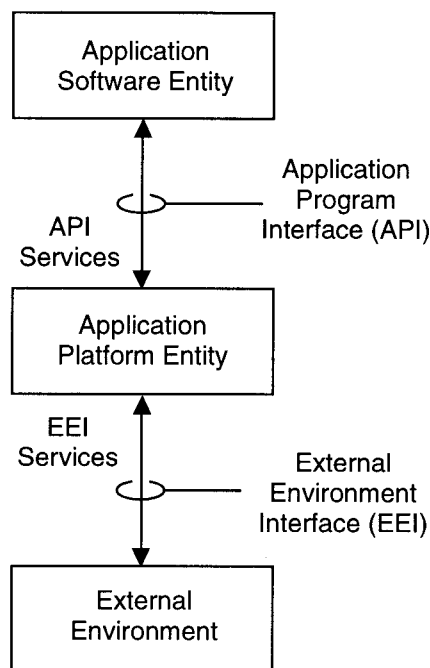


Figure 3.1—POSIX OSE Reference Model

The reference model is not a layered model, although the graphical representation may give that impression.¹⁵ It is the definitions of, and relationships among, the parts of the model that are important, rather than the graphical representation. Each of the parts interacts with the others across the interfaces, and no other relationship (for example dependence, importance, superiority, or primacy) is implied or intended.

The application platform provides services to a variety of users across both platform interfaces. The EEI is actually composed of three types of interfaces, all of which share the common characteristic of being externally visible. A more complete discussion of EEIs is found in 3.2.2.1. A human user invokes the platform services at the EEI. A programmer invokes application platform services at the API by writing source code, which accesses the service when compiled and executed.

All of these features may be available locally or remotely if the system is connected to a larger distributed system. All other resources can be conceptualized as being contained within the application platform.

Note that the actual implementation of any given system element may differ greatly from the reference model presented. The intention is to define a conceptual reference model that the widespread design, implementation, and integration communities may assume in executing their activities. Partitioning of function for purposes of discussion or specification does not imply or endorse similar partitioning for design or implementation.

3.2.1 Reference Model Entities and Elements

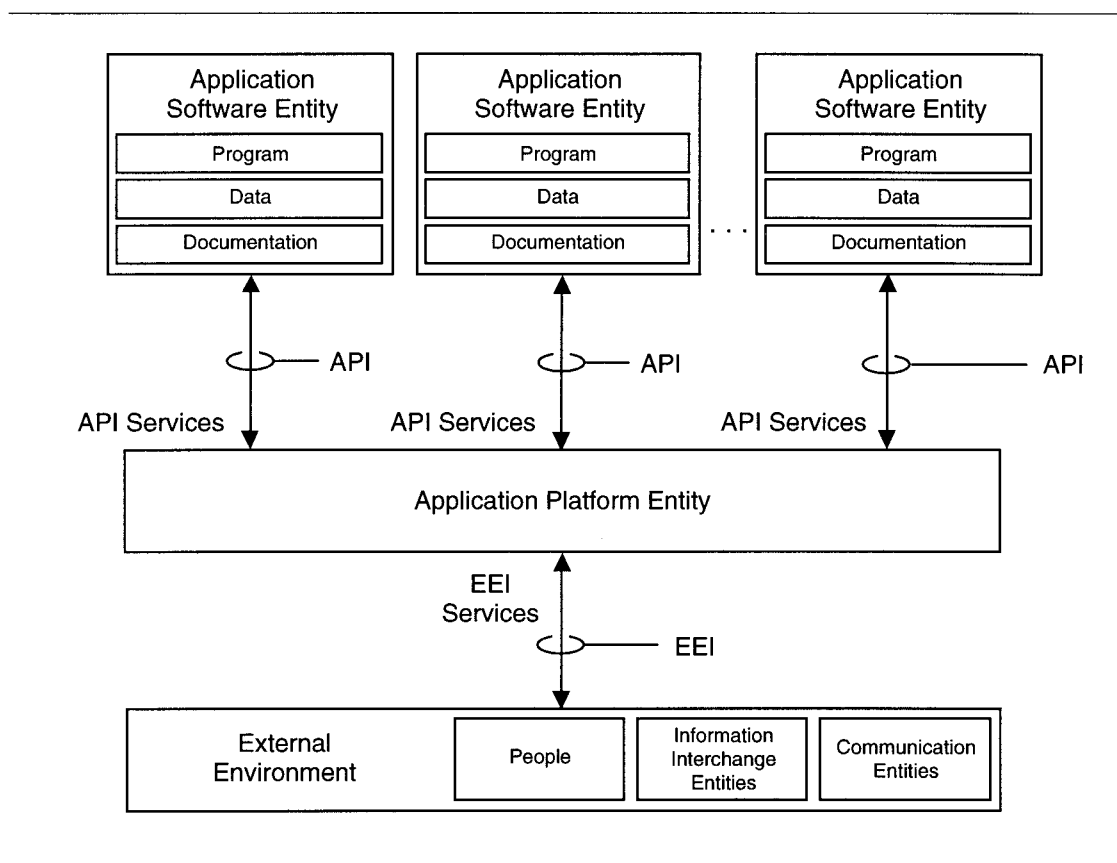


Figure 3.2—POSIX OSE Reference Model — Entities

¹⁵In fact, many other representations were considered (for example, various rotations, circular symmetry, etc.).

Figure 3.2 expands Figure 3.1 to identify elements of the reference model entities. For the purposes of this discussion, the term *entities* is used when discussing the classification of items related to application portability. The term *entity* is used in the dictionary sense, with no hierarchy implied. The only distinction needed in this discussion is the distinction between a *thing* (or *entity*), and an interface (or boundary between things).

Application software is defined (see 2.2.2.6) as software specific to an application. It is composed of one or more of

- Programs (source code, command/script files, etc.)
- Data (user data, application parameters, screen definitions, etc.)
- Documentation (online documentation only; hardcopy not included)

An application program is represented by source code, produced according to one or more specific programming languages and a set of language bindings (i.e., API specifications) for the required services. These specifications may be standards, public specifications (see 2.2.2.42), or proprietary specifications.

An application program may be divided into two parts:

- An *invariant* portion of source code, requiring no change when ported
- A *variant* portion of source code, which requires changes when ported

The objective of any effective application software portability method or project should be to minimize the variant portion of the application software via creation and use of API standards. This would ideally allow application software components to be moved to a different (but standards conforming) application platform and run without source code modification.

Separate, but related, standards may be required to support the portability of each of the elements listed above. Examples of application software are the familiar word-processing, spreadsheet, or accounting packages, as developed by the consumer or a commercial application software developer. Each of these packages appears as an application software entity when executed on an application platform.

One or more applications may run on a given application platform simultaneously, as represented by the boxes at the top of Figure 3.2. Each application can be thought of as an independent application entity, communicating and synchronizing with other applications, if necessary, via a variety of communication mechanisms.

The application platform is defined (see 2.2.2.4) as the set of resources that support the services on which an application or application software will run. It provides services at its interfaces that, as much as possible, make the implementation-specific characteristics of the platform transparent to the application software.

In order to assure system integrity and consistency, application software entities competing for application platform resources need to access all resources via service requests across the API.

The application platform concept does not imply or constrain any specific implementation beyond the basic requirement to supply services at the interfaces. For example, the platform might be a single processor shared by a group of applications, or it might be a large distributed system with each application dedicated to a single processor. (See 3.2.4.)

The application platform implementations that use the POSIX OSE may differ greatly depending upon the requirements of each system and its intended use. It is expected that application platforms defined to be consistent with the POSIX OSE will not necessarily provide all the features discussed here, but will use tailored subsets for a particular set of application software.

The external environment contains the external entities with which the application platform exchanges information. These entities are classified into the general categories of human users, information interchange entities, and communication entities.

Human users are not further classified, but are treated as typical persons. Information interchange entities are physical data storage media, including, for example, CD-ROM, magnetic tapes, floppy disks, and security badges. Communication entities include phone lines, local area networks, and packet switching equipment.

3.2.2 Reference Model Interfaces

Figure 3.3 expands on Figure 3.1 to identify categories of services available at the reference model interfaces. Between the general model entities there are two types of interfaces, labeled as the API and the EEI.

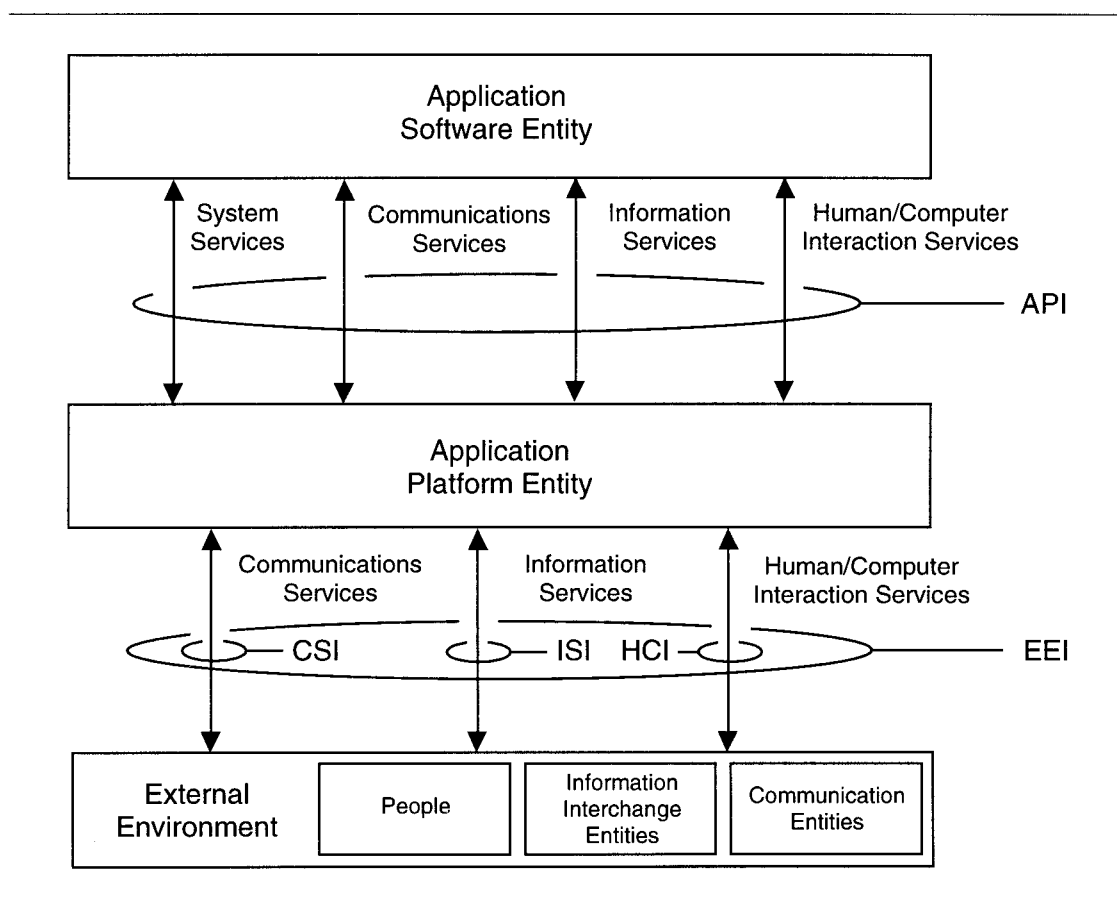


Figure 3.3—POSIX OSE Reference Model — Interfaces

3.2.2.1 EEI

The EEI is defined (see 2.2.2.14) as the interface between the application platform and the external environment across which information is exchanged. It is defined primarily in support of system and application software interoperability.

User and data portability are directly provided by the EEI, but application software portability is indirectly supported by reference to common concepts linking specifications at both interfaces.

An example of a common concept linking both the human/computer interaction API and the human/computer interaction EEI is a “window.” At the API, a hypothetical service

```
Open_Window(x1, y1, x2, y2)
```

could be specified to draw a rectangular region on a display, with its upper left corner at screen coordinates (x1, y1) and its lower right corner at screen coordinates (x2, y2). At the EEI, conventions for the appearance and behavior of a window may be documented independently of the API. For instance, a convention may be adopted that all windows will have borders and "handles" on the corners that a user can use to resize the window.

Changes to the specifications for the API and EEI can be made independently to some degree, but the concept of a window is fundamental to both. To achieve portability for these services, a common concept for a window has to be adopted across the set of platforms for which portability is desired. Other common concepts are associated with each of the external interfaces.

The following three interfaces are EEI type interfaces:

- HCI
- ISI
- CSI

The HCI is the boundary across which physical interaction between the human user and the application platform takes place. Examples of this type of interface include CRT displays, keyboards, mice (or other pointing devices), and audio I/O devices. Standardization at this interface will allow users to access the services of POSIX OSE-based systems without costly retraining.

The ISI defines a boundary across which external, persistent storage service is provided, where only the format and syntax are required to be specified for data portability and interoperability.

The CSI provides access to services for interaction between internal application software entities and application platform external entities, such as application software entities on other application platforms, external data transport facilities, and devices. The services provided are those where protocol state, syntax, and format all should be standardized for application interoperability.

3.2.2.2 API

The API is defined (see 2.2.2.5) as the interface between the application software and the application platform across which all services are provided. The API is defined primarily in support of application portability, but system and application software interoperability also are supported via the communication services and information services APIs.

The POSIX OSE API is a combination of a number of standards-based interfaces. It can be thought of as a bookshelf containing several standards-based APIs, with each API a separate book on the bookshelf.

The POSIX OSE API describes a complete set of services provided at the interface between the application software and the underlying application platform. The services provided at the API may be divided into the following categories:

- System Services, including core system services and language services
- Communication Services
- Information Services, including database services, data interchange services, and transaction processing services
- Human/Computer Interaction Services, including user command interface services, character-based user interface services, windowing system services, graphics services, and application software development support services

The first API grouping provides access to services associated with resources that are internal to the application platform. The last three API groupings provide the application software with access to services associated with external environment entities.

These four categories of API services describe the range of function needed at the API. The specifications (that is, the documents themselves) may take the form of programming language specifications, language-independent service specifications, and language bindings for the service specifications. These specifications may be described as follows:

- Specifications traditionally associated with languages, such as program control (if ... then ... else), math functions, string manipulation, concurrency, low-level programming mechanisms, etc., are defined as *the programming language API specifications*.
- Specifications for services provided by the underlying application platform defined independent of any programming language, such as interprocess communications, interobject messages, access to the user interface, and data storage, are identified as *language-independent service specifications*.
- Language-independent service specifications are translated into language-specific specifications used by programmers in writing applications. These specifications provide access to the services using methods consistent with a specific programming language. Such language-specific specifications are called *language-binding API specifications*.

While it is acknowledged that language-independent specifications are not yet common, their creation is expected to facilitate the management and development of consistent language-binding standards. The language-binding specifications are used directly by programmers and application platform suppliers in implementing application software and platforms.

The “programming language”/“language binding” dichotomy may be a result of the way information technology standards are currently developed. Programming language specifications are developed with the goal of being “system independent” (e.g., C, COBOL, Fortran, etc.). Language-binding specifications (e.g., ISO/IEC 9945-1 :1990 {68}, MOSI, GKS, SQL, etc.) are being translated into “language-independent” specifications, with one or more bindings for specific languages. Revisiting the “bookshelf” concept described at the beginning of the subclause, the following question can now be answered: “What books (standards) does the programmer need to produce application software source code that is fully portable?” Once the programmer decides (or is told) which language to use, the programmer needs the programming language manual (e.g., C or Ada) and the language-binding specifications that provide all the required services (e.g., C bindings for graphics, networking, and indexed files). In this context, it is clear that the base language specification itself is a major component of the API. In fact, many useful programs can be written using only the base language specifications.

3.2.3 EEI-API Service Relationships

The relationships between services provided at the API and similarly named services at the EEI are not simple one-to-one relationships. For example, a data storage service interface may provide an application with transparent access to a remote file via network services. In this case, the completion of the data storage service provided at the API is dependent upon (“translated” into) communication services provided at the EEI.

In the general case, application software entities never access the EEI directly, although service requests at the API will often result in service invocations at the EEI. This usually happens using mechanisms that the application programmer never observes or cares about, as long as the service request is satisfied. Likewise, the information services that are accessible at the API may sometimes be performed internally, without invoking an EEI function.

Fortunately, it is not essential for the purpose of satisfying the requirements of the POSIX OSE to specify these relationships in detail. In fact, too detailed a definition could unnecessarily constrain the implementation. A given implementation of the application platform will define the relationship between the API and EEI in different ways.

3.2.4 POSIX OSE-Based Distributed Systems

In a distributed environment, multiple application platforms may interact by way of a communication mechanism external to the platforms. Application platforms interact with the communication services EEI, as in Figure 3.4. When an application software entity requests communication with another entity on a different platform, the request is made

at the API. The implementation of the application platform translates these API requests into appropriate actions at the EEI.

Communication occurs between application platforms via external entities that implement the data transport function at the communications services EEI. These can use a wide variety of implementation methods and protocols, providing access to distributed data and services via a network.

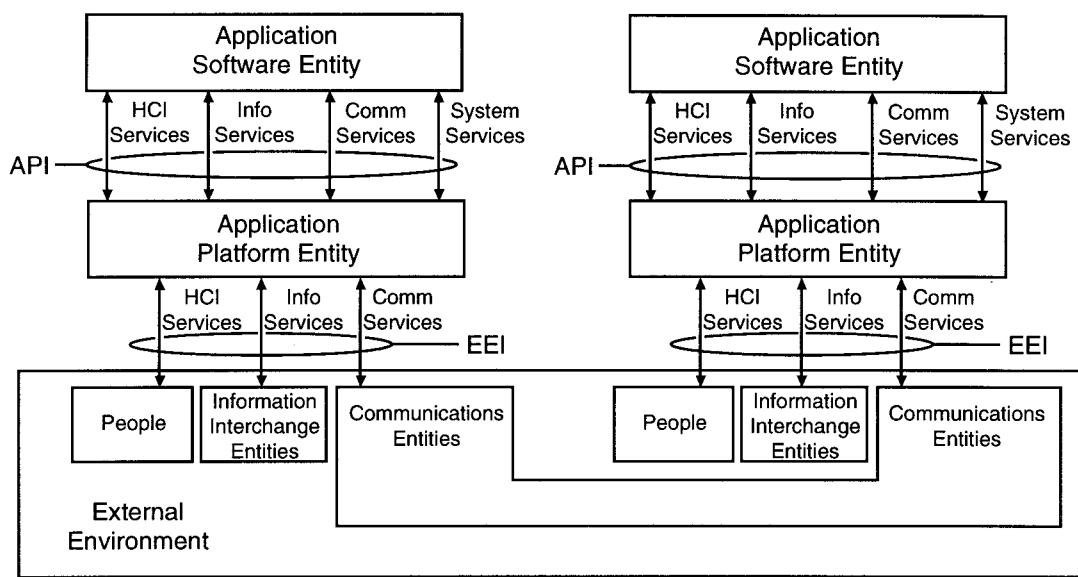


Figure 3.4—POSIX OSE Reference Model — Distributed Systems

This treatment of the communications services EEI is also used to support the implementation transparency of the application platform. Figure 3.5 sketches an application platform implementation where the platform is actually a distributed system composed of multiple platforms. In this scenario, the application platform is labeled as a “perceived” platform. This distinction is important, for example, during procurement. The procuring authority specifies requirements of a perceived platform, and the bidders respond with proposals that describe alternative implementations of the platform. One proposal could implement the platform using a single monolithic processor, such as a classic mainframe. Another proposal might provide a variety of specialized processors connected by a network. A wide variety of alternatives is possible, each with its own advantages and disadvantages. By specifying the procurement requirements as a perceived platform, the buyer can consider the widest range of alternative implementations and more easily tailor the solution to fit the needs of the buyer.

Note that the treatment of distributed systems within this guide is limited to those aspects directly related to application portability and interoperability. A complete treatment of distributed systems would require a considerably expanded discussion, which is beyond the scope of this guide.

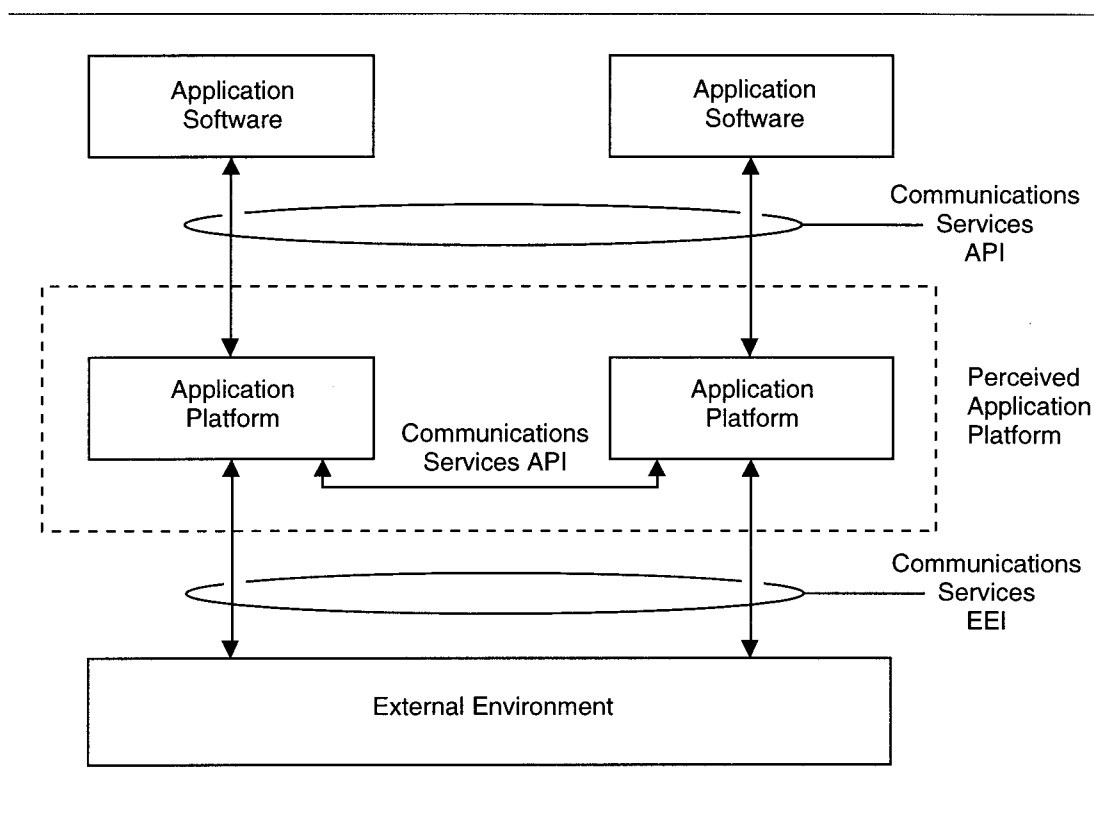


Figure 3.5—Distributed Application Platform Implementation

3.3 POSIX OSE Services

Applications may use services that are distributed among a number of different computers. This implies that one application may use many POSIX services. This approach becomes even more valuable when a distributed environment has to be designed to support many applications using a single resource like a database. Note that the database itself may be implemented across multiple machines. Similarly, the notion of cooperative work will eventually require support of many users accessing a single application that arbitrates both screen and database usage.

This guide describes services provided to users of application platforms in support of POSIX objectives of application portability and system interoperability. These services are available to users across specified interfaces keyed to the POSIX OSE reference model defined in 3.2.

The POSIX OSE services are divided into categories described by the clauses in Section 4. Each category begins by defining a more detailed and specialized version of the POSIX OSE reference model to provide context for service specification. Services and associated standards are then described for each category. Finally, POSIX OSE cross-category services affecting each category are discussed.

Further refinement, based on evolving standards and technology, will lead to further evolution of this guide.

3.4 POSIX OSE Standards

The identification of a complete, consistent suite of standards for the POSIX OSE will, by necessity, draw from many forums. One of the criteria for judging completeness is the satisfaction of the full range of services required from the application platform. The factors used to select standards will be described, followed by the selection precedence.

Although the services are stated with a clear partitioning in mind, the standards reflect existing partitioning. These standards were created within disparate organizations and projects, which were in many cases carried out in isolation from the others. As a result, mapping of services to standards is not a simple relationship.

3.4.1 Factors in Standards Selection

Substantial investment of time and resource is made after an organization commits itself to a specification, and such commitment should be carefully considered. The following concepts identify some of the criteria used to select standards for inclusion in the POSIX OSE. Other organizations may find these criteria useful as well, in situations where no clear selection is available, and judgment is required.

3.4.1.1 Openness

Specification development organizations can differ from one another by virtue of their openness. The fewer the barriers to participation and the greater the number of constituencies represented, the more open is the forum. The result is a varying degree of consensus in the technical content of the standards across development bodies.

As a general rule, standards developed by accredited standards development organizations (all of which use an open forum) are preferred over those specifications developed by bodies using a closed forum.

3.4.1.2 Stage of Development

There is a standards development life cycle process whose effects need to be taken into account. Most standards follow a sequence from approved development, through draft, and on to approved standard.

As a general rule, where choices are made among standards, those approved or closer to approval are favored.

3.4.1.3 Stability

The concept of stability refers to anticipated change in the standard over time. This change may expand or contract the technical coverage of the standard.

As a general rule, the more stable standards are preferred over those subject to change. In some cases, however, stable standards are not listed, as in the case of obsolete or not widely implemented standards.

3.4.1.4 Geographic Scope of Consensus

There are differences among standards development bodies with respect to the scope of their geographic consensus. Accredited standards development bodies are typically empowered to develop standards for the international, regional, or national level.

The general rule applied in the selection of standards for inclusion in the POSIX OSE is to select standards developed by those bodies that have the greatest scope of coverage. This results in a precedence for standards selection of international, followed by regional, followed by national body standards.

3.4.1.5 Functional Scope Addressed Within This Guide

A specification is listed only if it addresses some service requirement listed in this guide. Standards and/or specifications listed are not, however, limited to one set of services.

3.4.1.6 Consistency With ISO/IEC 9945-1:1990 {68}

Standards listed in this guide are suitable for inclusion in a profile with ISO/IEC 9945 -1:1990 {68}. Standards listed in this guide are intended to be consistent with ISO/IEC 9945 -1:1990 {68}. Selection for inclusion in this guide, however, does not guarantee that this condition is met.

3.4.1.7 Availability for Unencumbered Implementation

This guide emphasizes those specifications that may be used, without encumbrance, to implement a conforming application platform. The specification qualifies for inclusion in this guide even if the document itself is a salable item.

3.4.2 Selection Precedence

3.4.2.1 Selection Precedence for Standards

The following list shows the selection precedence of standards cited in this guide. The order from top to bottom is from most to least preferred.

- 1) Approved standards maintained by accredited international standards development organizations
- 2) Approved standards developed by accredited regional bodies
- 3) Approved standards developed by accredited national bodies
- 4) Draft standards developed by accredited international bodies

ISO/IEC Joint Technical Committee 1 (JTC 1) rules allow the citation of Draft International Standards (DISs) as normative references in International Standards and, thus, they are considered part of the POSIX OSE. This does not include working group Committee Documents (CDs), or any draft regional or draft national standards, which are considered emerging standards.

A forum qualifies as *accredited* if it operates under the authority of ISO, IEC, ISO/IEC JTC 1, or ITU-T. This includes those national body forums and regional workshops sanctioned to participate directly in these international forums.

3.4.2.2 Selection Precedence for Specifications Other Than Standards

Where specifications other than standards are discussed in the context of temporary measures to address gaps in available standards, the following order of precedence is recommended for use in selecting among candidate specifications where multiple suitable specifications are available:

- 1) Draft standards developed by accredited regional bodies
- 2) Draft standards developed by accredited national bodies
- 3) Approved (as opposed to draft) specifications (widely adopted, but not formal standards) developed or maintained or both in an open forum, including consortia specifications or certain de facto “standards”
- 4) Specifications developed by a closed forum

A de facto standard is one that has been widely accepted in practical use by users. This may include widely used formal standards, consortia documents, or product offerings. A useful form of specification is one that is both a de facto and a formal standard.

Standards projects for which there is no draft or approved standard are not included in the POSIX OSE.

Only the highest precedence specification is listed or discussed in the main text.

This guide only cites government and de facto standards and specifications in discussion of gaps in available standards.

3.5 POSIX Profiles

The results of open system standardization projects contribute to an expanding set of *base standards*, addressing a growing subset of functional requirements.

Profile projects then select among these base standards to create a tailored, consistent set of standards addressing a more specific type of system or set of application software. Profiles satisfy the requirements of application *domains* such as office or industrial automation, transaction processing, or realtime control systems.

The POSIX OSE model provides a way to characterize the functionality of profile activities. The current OSI profiles tend to focus strictly on communication services at the EEI. Other profiles might focus on a single component or span multiple interface types.

3.6 PIIs

This guide treats the application platform as a single monolithic combination of hardware and software that supports application portability and application interoperability. In some models or architectures it may be necessary to show some of the internal details of how the application platform is constructed out of various service components made up of either hardware or software.

In this model, the interfaces between service components that make up the platform are called PII. Figure 3.6 shows how a PII can be represented in the model.

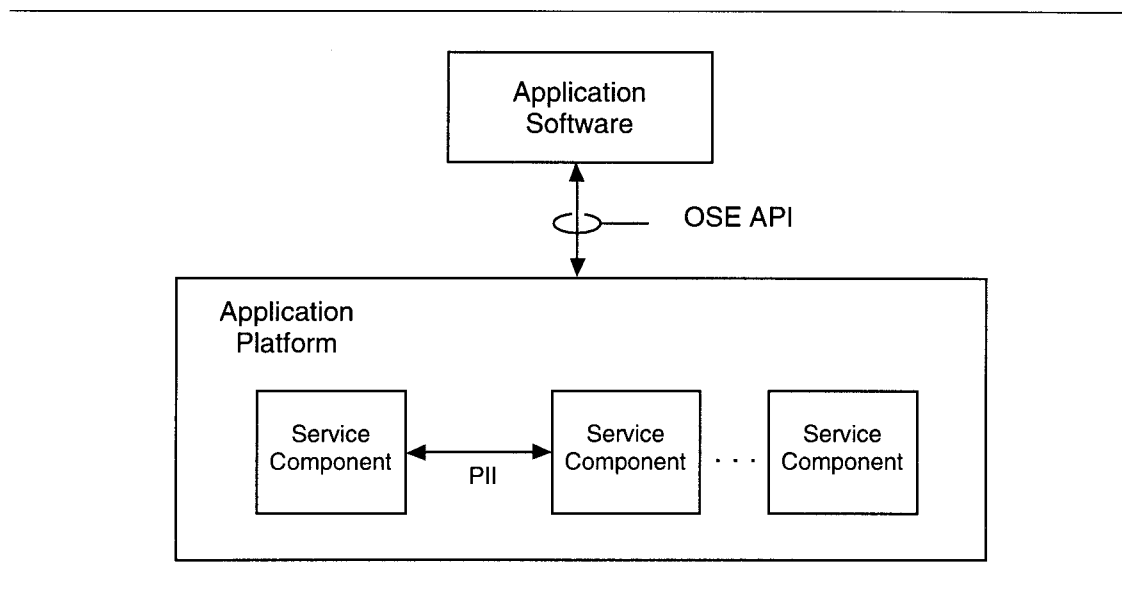


Figure 3.6—Service Components and Interfaces

There are a number of important interfaces within the application platform that are important for understanding how computer systems can be assembled. These interfaces include

- Interfaces between the various software components of an operating system
- Interfaces between the operating system and hardware
- Backplane interfaces
- I/O device interfaces

The POSIX OSE reference model does not attempt to identify or apply standards to this type of interface. These interfaces are considered to be outside the scope of this guide because they

- Do not affect application portability and interoperability
- Do not affect specification of the API and the EEI
- Are driven primarily by a specific technical approach to implementing an application platform

Specification of PII's in this guide would represent an unnecessary constraint on the implementation of the application platform, and are unnecessary for the specification of the API and the EEI.

4. POSIX OSE Services

This section describes the services that support the objectives identified in this guide. The four major service categories are partitioned into a more detailed set of subcategories. This partitioning is illustrated in Table 4.1.

Table 4.1—Mapping of Service Categories to Section 4 Clauses

Service Category	Clause	Subcategories
System	4.1	Language Services
	4.2	Core System Services
Communications	4.3	Communications Services
Information	4.4	Database Services
	4.5	Data Interchange Services
	4.6	Transaction Processing Services
Human/Computer Interaction	4.7	User Command Interface Services
	4.8	Character-Based User Interface Services
	4.9	Windowing System Services
	4.10	Graphics Services
	4.11	Application Software Development Support Services

Criteria used to partition services are outlined in 3.2 and discussed at the beginning of each clause. The discussion for each of the service category subclauses follows the same outline, and is as follows:

- 4.n.1 Overview and Rationale
This text gives an overview of the service category and rationale for its use as a category.
- 4.n.2 Scope
This text introduces the scope of this service category and the criteria used to identify the services within it.
- 4.n.3 Reference Model
This subclause builds on the model of 3.2 and gives additional details related to the interfaces and services discussed. An optional subclause may discuss implementation considerations, similar to the discussion of 3.6
- 4.n.4 Services
This text provides the definition of services within the scope described in 4.n.2.

4.n.5 Standards, Specifications, and Gaps

A table lists the standards and specifications available to meet the services listed in 4.n.4. This is followed by a brief discussion of services for which standards are not available. The list of standards in the table is intended to be comprehensive for the area covered by the 4.n.4. services; there are no applicable standards (those recognized by ISO as legitimate normative references) excluded from the POSIX OSE. Within the table, the Type column refers to the status of the standard:

S	A current standard
E	An emerging standard
P	A public specification
G	An unaddressed service (gap)

The lines marked E, P, and G are shaded to highlight their status as outside the current POSIX OSE.

4.n.5.1 Standards in the POSIX OSE

This subclause lists the standards considered to be part of the POSIX OSE—existing specifications that have been approved as standards by accredited standards bodies, in the order of precedence identified in 3.4.2. When services are satisfied at a higher precedence level, comparable specifications at a lower level are not listed.

4.n.5.2 Additional Specifications

4.n.5.2.1 Emerging Standards

This subclause provides an alphabetized list of specifications and/or activities that address the functional areas within the 4.n section, but which have not yet been completed. Where a group or activity is cited, the charter of the group may address the functionality, but it is possible that a draft may not be available. Only those services not currently addressed by existing standards are discussed in this subclause. It is expected that documents will migrate from 4.n.5.2.1 to 4.n.5.1 as they complete the consensus process and this guide is updated.

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully. There may be some risk in using these emerging standards prior to their final approval. These risks need to be balanced by the risk involved in continued reliance on nonstandard or proprietary solutions.

4.n.5.2.2 Public Specifications

This subclause lists any specification outside of the formal standards community that is available to anyone (e.g., no membership required) for implementation and distribution (including sale), preferably without restriction, including government, consortia, and de facto standards, in accordance with the selection factors cited in 3.4.1.

The specifications listed in this subclause are not part of the POSIX OSE, but may be of interest. Use of these specifications should be considered carefully. There may be some risk in using these emerging standards prior to their final approval. There is also risk because such specifications are not approved by an accredited standards development organization. These risks need to be balanced by the risk involved in continued reliance on nonstandard or proprietary solutions. They are included in this guide to indicate some of the existing work that has been done in areas that are gaps in the POSIX OSE.

4.n.5.3 Unaddressed Services

This subclause lists the services for which no specification has been cited in this guide. Products may be cited here to illustrate capabilities that are not addressed by standards.

4.n.6 POSIX OSE Cross-Category Services

This subclause contains discussion of the cross-category services in Section 5 that are specific to clause 4.n.

4.n.7 Related Standards

This subclause is optional and may identify interdependencies among standards that should be taken into account when selecting among them.

4.n.8 Open Issues

This subclause is optional and may identify issues under discussion in the open systems community.

4.1 Language Services

4.1.1 Overview and Rationale

While a consistent interface to the operating system is essential for applications portability, the application will have been developed using language and system development tools that, in turn, require support by standards to achieve source code portability.

As discussed in 3.2.2.2, a programmer wishing to write portable application software source code will use one or more standardized programming languages. By reference to the relevant programming language manual and language binding specifications, the programmer will be able to acquire any necessary services from the application platform. It is therefore appropriate to identify programming languages as elements of an OSE. The choice of language is influenced by several factors including functionality, suitability for the proposed application, existing in-house skills, or the management practices of an organization.

Confidence in the portability of a software component is enhanced by the knowledge that it has been written in a language supported by an international standard and the code compiled using a compiler that has a certificate of conformance issued by an accredited test center. Nonconforming extensions must be avoided if applications portability is to be maintained.

The languages that have been identified in this document are those seen to be in most popular use today for software development. The ISO/IEC 9945 -2:1993 {69} shell command language and such utilities as awk are considered in 4.7. The standards identified are the most widely recognized today, with significant use in the information technology industry on a broad range of processors, or where a large installed base of a particular version is known to exist.

4.1.2 Scope

The services described in this clause cover the most widely used unstructured languages (such as COBOL and Fortran) and block-structured languages (such as C and Pascal) that are currently in use for the development of applications; i.e., the languages used to write application programs. These are referred to commonly as third-generation languages. The fourth-generation languages that evolved from languages such as COBOL and Fortran are not addressed in this guide.¹⁶ In order for a program to address an API to the services described in other clauses of this guide, an appropriate language binding to that interface is required. References to those bindings will be found in the clause describing the relevant service.

4.1.3 Reference Model

This subclause identifies the entities and interfaces supporting language services. The reference model for language services is shown in Figure 4.1, based on the POSIX OSE reference model in Figure 3.1. The language services are required to support the binding of the applications at the API. They are not visible at the EEI. However, the external environment is shown in Figure 4.1 for completeness.

At a simplistic level, the programmer developing an application that requires access to only basic operating system services will use a compiler that meets both the fundamental language standard (e.g., ISO 1989 :1985 {3} for COBOL, ISO 1539 :1991 {2} for Fortran) and any bindings established for the relevant APIs in ISO/IEC 9945 -1:1990 {68}.

¹⁶Such fourth-generation languages may form part of future POSIX standardization efforts.

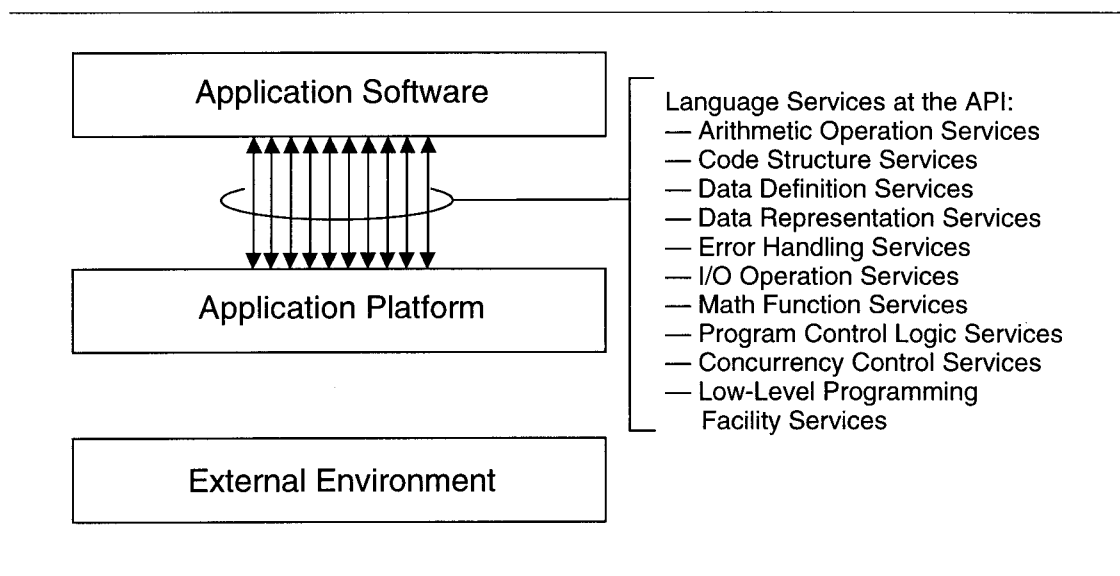


Figure 4.1—POSIX OSE Language Services Reference Model

An application program may also require other information technology services, such as database services, graphics services, and networking services. In such cases, the vendors of these services will offer an API to meet the requirements for the popular programming languages.

The POSIX OSE includes support for one or more of the languages identified in 4.1.4.

4.1.4 Services

Language services provide the basic syntax and semantic definition for use by a software developer to describe the desired application software function. Many language services are a unique function of the language specification. The details of these languages are found in the relevant language manuals and supporting standards.

4.1.4.1 API Services

Programmers require the ability to write and execute a program in a particular programming language. The selection of a particular programming language for the development of an application depends on the ability of the language to provide some or all of the functions listed here, as well as some specialized capabilities unique to the language:

- Arithmetic operation
- Code structure
- Concurrency control
- Data definition
- Data representation
- Error handling
- I/O operations
- Low-level programming facilities
- Mathematical functions
- Program control logic

Such services as concurrency control, program control logic, and data declarations are important elements of the OSE. They specify the notation in which the application program is written and are used to invoke execution control, sequencing, and storage allocation services at the API.

The programming languages identified in this clause are

- Ada
- APL
- Full Basic
- C
- C++
- COBOL
- Common Lisp
- Fortran
- Modula-2
- Pascal
- PL/I
- Prolog

In addition to referring to the relevant language standard, if a programmer requires other services (e.g., the Graphics Kernel System), it is necessary to refer to the relevant language binding to those services. Language bindings are identified in the Standards subclause, 4.n.5.2, of each service clause in Section 4

4.1.4.1.1 Ada

Ada is a procedural language capable of processing both numerical and textual data that has the key attributes of

- Strong data typing
- Data abstraction
- Structured constructs
- Multitasking
- Concurrent processing
- Support for object-oriented programming

Although Ada was developed initially for military purposes, it is considered suitable for a variety of business and industrial applications, particularly very large applications.

4.1.4.1.2 APL

APL is a language and interactive programming environment oriented around multidimensional arrays of characters and numbers. It uses an extremely compact notation based on powerful primitive functions and function-combining operators. Revisions to the language are in preparation to permit single-array elements to contain multiple arrays. APL has an established user base in financial analysis.

4.1.4.1.3 Full Basic

Full Basic is a procedural language often implemented in an interactive form. It has some similarity to Fortran. It is readily learned by non-computer-literate individuals. Commonly used for educational purposes, it has also been adopted in a variety of business and commercial applications running on small business systems. Full Basic offers

- Conversational statements
- Free style input
- Segmentation of complex statements
- Mathematical functions

4.1.4.1.4 C

C is a general-purpose procedural language that was developed for the UNIX operating system. It offers the control and data structure of a high-level language and the efficiency of primitive operators that have made it very suitable for system programming.

4.1.4.1.5 C++

C++ has evolved as a superset of C and may be viewed as a procedural language, while at the same time offering the capability for object-oriented programming. The concept of an object-oriented language is to define data objects that include sets of operations that can manipulate the data, and so to direct these objects to apply the necessary operations that comprise the application.

4.1.4.1.6 COBOL

COBOL is a procedural language designed originally to meet the needs of business. It permits use of natural words and phrases, enabling the language to be adopted by nontechnical writers with a basic appreciation of information processing. The language offers file organization features, variable data length, input/output procedures, indexed file access, and report generation.

4.1.4.1.7 Common Lisp

Lisp is an interactive functional language. The basic entity is the symbolic expression, which is either an atomic symbol or a list structure. A list is a set of items in a specific order. Lists can be of variable lengths and dynamically adjusted; the items can be of different types. The language is based on high-level logic constructs and used in artificial intelligence applications.

4.1.4.1.8 Fortran

Although originally developed for processing scientific problems, Fortran is widely used in commercial and educational applications. It is a procedural language whose grammar, symbols, rules, and syntax follow simple mathematical and English-language conventions. Its focus is on numerical computation, using simple, concise statements, and it operates on small amounts of input data and little text.

4.1.4.1.9 Modula-2

Modula-2 is a general-purpose procedural language based on Pascal, but extended by

- The module concept and separate compilation
- More convenient syntax and stronger typing
- Access to machine level facilities
- Concurrent programming

4.1.4.1.10 Pascal

Pascal is a procedural language that is particularly effective in structured programming and that was designed to help programmers in rapid error detection. It is highly convenient, handling both numerical and textual data. It is considered suitable for small system applications such as typesetting, editorial work, computer-aided design, and manufacturing processes.

4.1.4.1.11 PL/I

PL/I is a procedural language introduced to offer in one language the strengths of both COBOL and Fortran; i.e., serving both the business and scientific communities. It has the Fortran strength of simple statements, coupled with the

ability, as in COBOL, to manipulate data and organize files. It is block structured, facilitating good programming techniques.

4.1.4.1.12 Prolog

Prolog is a nonprocedural language that has an emphasis on description rather than on action. It is described as pattern-directed rule-based programming using definitions of conditions established within the program to satisfy a query. It is of particular value in applications of artificial intelligence and for constructing expert or knowledge-based systems.

4.1.4.2 EEI Services

Not applicable.

4.1.5 Standards, Specifications, and Gaps

See Table 4.2.

Table 4.2—Language Standards

Service	Type	Specification	Subclause
Ada	S	ISO 8652 :1987 {33}	4.1.5.1
APL	S	ISO 8485 :1989 {24}	4.1.5.1
Full Basic	S	ISO/IEC 10279:1991 {82}	4.1.5.1
C	S	ISO/IEC 9899 :1990 {67}	4.1.5.1
C++	E	n/a	4.1.5.2.1
COBOL	S	ISO 1989 :1985 {3}	4.1.5.1
Common Lisp	E	n/a	4.1.5.2.1
Fortran	S	ISO 1539 :1991 {2}	4.1.5.1
Modula-2	S	ISO/IEC DIS 10514 {85}	4.1.5.1
Pascal	S	ISO/IEC 7185 :1990 {16}, ISO/IEC 10206 :1991 {81}	4.1.5.1
PL/I	S	ISO 6160 :1979 {11}	4.1.5.1
PL/I (GP Subset)	S	ISO/IEC 6522 :1992 {13}	4.1.5.1
Prolog	S	ISO/IEC 13211-1 :1995 {98}	4.1.5.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.1.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.1.5.1.1 Ada

ISO/IEC8652 :1987 {33} is the current version of the international standard for Ada, often referred to as Ada 95. However, there are many application programs still in service that were developed using Ada-83 (ISO/IEC8652 : 1982). It is therefore acceptable to use Ada-83 to facilitate transition to an OSE, particularly since the Ada-83 binding to ISO/IEC9945-1 :1990 {68} has become IEEE Std 1003.5-1992 {119}.

4.1.5.1.2 APL

ISO 8485 :1989 {24} is the current version of the international standard for APL.

4.1.5.1.3 Full Basic

ISO/IEC 10279 :1991 {82} is the current version of the international standard for Full Basic.

4.1.5.1.4 C

ISO/IEC 9899 :1990 {67} is the current version of the international standard for the C language.

4.1.5.1.5 COBOL

ISO 1989 :1985 {3} is the current version of the international standard for COBOL, which is an endorsement of ANSI X3.23-1985. Addendum 1 to the standard, on “intrinsic functions,” has been published.

4.1.5.1.6 Fortran

ISO 1539 :1991 {2} is the current version of the international standard for Fortran. However, there will be many application programs still in service that were developed using FORTRAN-77 (ANSI X3.9-1978 {103}), and it is acceptable to use this version to facilitate transition to an OSE, particularly since the FORTRAN-77 binding to ISO/IEC 9945-1 :1990 {68} has become IEEE Std 1003.9-1992 {120}.

4.1.5.1.7 Modula-2

ISO/IEC DIS 10514 {85} defines the Modula-2 language formally, using the Vienna Definition Method (VDM) and specifying library modules. In addition, SC22/WG15 has proposed a new work item to specify a Modula-2 binding to ISO/IEC9945-1 :1990 {68}.

4.1.5.1.8 Pascal

ISO/IEC7185 :1990 {16} is the current version of the international standard for Pascal. The international standard for Extended Pascal is ISO/IEC10206 :1991 {81}.

4.1.5.1.9 PL/I

ISO 6160 :1979 {11} is the current version of the international standard for PL/I, which is an endorsement of ANSI X3.53-1976. ISO/IEC 6522 :1992 {13} is the current version of the international standard for a general-purpose subset of PL/I, which is an endorsement of ANSI X3.74-1987. A revision of this standard is at the DIS stage.

4.1.5.1.10 Prolog

ISO/IEC 13211-1 :1995 {98} is the current version of Prolog.

4.1.5.2 Additional Specifications

4.1.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.1.5.2.2 C++

ISO/IEC JTC 1/SC22/WG21 has a work item for standardizing C++. This will be based on a standard under development in ASC X3J16.

4.1.5.2.3 Common Lisp

ISO/IEC JTC 1/SC22/WG16 has a work item for standardizing Common Lisp. This will be based on a standard under development in ASC X3J13 (X3.226).

4.1.5.2.4 Public Specifications

The X/Open COBOL specification {B64} is aligned with Amendment 1 to ISO1989 :1985 {3}, with the addition of definitions for file locking, screen handling, and internationalization.

4.1.5.3 Unaddressed Services

The installed base of corporate mission-critical application software and its application platform is founded upon the use of COBOL and the database management systems available on the application platform. There is a growing need for a new work item to define a COBOL language binding for ISO/IEC 9945-1 :1990 {68}.

4.1.6 POSIX OSE Cross-Category Services

Not applicable.

4.1.7 Related Standards

Many of the services within the POSIX OSE require APIs with bindings to languages identified in this clause; e.g., graphics and database. Reference to the particular language binding standard is in the relevant service clause.

4.1.8 Open Issues

While there are occasional calls for fourth-generation language standards, very few have been produced at this time.

There is a growing use of Smalltalk by application developers for object-oriented applications. Considerable improvement in development time through its use has been claimed. There is an interest in the standardization of Smalltalk.

Similarly there is interest in including MUMPS in future editions of this guide. MUMPS, using B-tree structures, is efficient at handling sparse arrays, ensuring both efficient occupancy and quick response times. It is an ANSI standard, ANSI/MDC X11.1-1990 {B7}. It has progressed internationally as ISO/IEC11756 :1992 {B3}.

4.2 Core System Services

4.2.1 Overview and Rationale

This clause describes the core system services component of the application platform. It presents a reference model for this component and describes the services provided to application software. Core system services are those usually considered part of an operating system or operating system executive, along with those services that may be provided to the application by system-level entities such as device drivers. Standards (current and emerging) that specify the APIs to those core system services are also described.

A common set of core system services provides support for the portability and the interoperability of application software. (Other services—for example, communications services in 4.3—are needed to support portability and interoperability fully). While other common services can aid application reuse, core system services are those that are common to the largest number of applications.

4.2.2 Scope

Core system services are those platform services that are responsible for the management of platform resources, including the processor, memory, files, and input/output. They generally shield applications from the implementation details of the machine. Core system services cover the areas of process management, file management, input/output, and memory management. Because there are a wide variety of platform users, ranging from large, general-purpose, time-shared systems to small, time-critical, special-purpose systems, services such as timers and clocks, event management, and logical device drivers have been included. Services related to distributed systems are also discussed, since application software sees these capabilities through the platform.

NOTE — A more thorough discussion of distributed systems has been deferred to a later version of this guide.

4.2.3 Reference Model

This subclause identifies the entities and APIs specific to the core system services of the POSIX OSE. The reference model presented here is consistent with and expands upon the reference model in Section 3. It provides the context for the discussion of core system services in this clause. The basic core system services model is shown in Figure 4.2.

Since the purpose of the core system services is to access the application platform resources, there is no unique EEI component of the core system services. At times, the core system services may make use of the EEI component of the other services. For example, to access a file transparently over a network, the core system services make use of the communication services EEI. Given that the core system services rely on the other service categories to provide EEIs, discussion of the EEIs is beyond the scope of this clause.

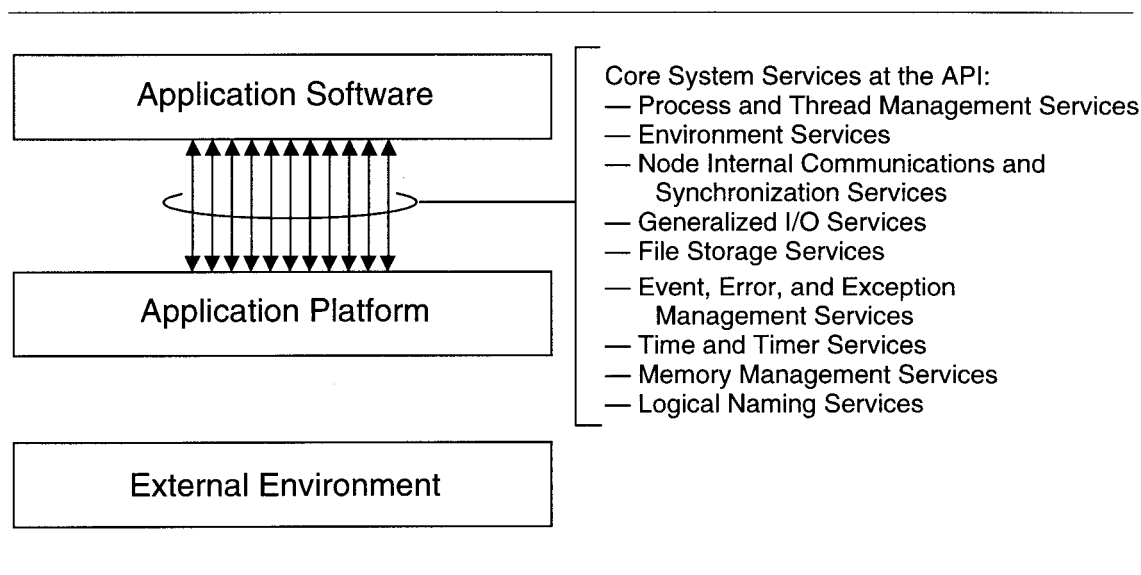


Figure 4.2—POSIX OSE Core System Services Reference Model

Most of the core system services may be available on either a single-system application platform or a multiple-system, distributed application platform. Some of the system services may be performed remotely if the system is a distributed system with multiple processor nodes. Such distribution is not illustrated in Figure 4.2 because it does not affect the API.

4.2.4 Services

This subclause identifies core system services that support application portability and system interoperability. The services directly available to an application program via the core system services API are addressed in 4.2.4.1. Further use of other service EELs for core system services is addressed in 4.2.4.2.

This subclause describes the major groups of system services that an application may require of an application platform. Not all of these services require a programming interface; therefore, services are described as either explicit or implicit services. Explicit services are those that can be accessed from an application program (via the API) and generally are only provided when requested. Implicit services, on the other hand, are services that the platform provides without a direct request. An example of an implicit service is the prevention of one program from writing over the memory of another. An example of an explicit service is a call to write the contents of a block of memory to some device.

4.2.4.1 API Services

This subclause describes the major categories of system services available at the core system services API. These services include

- Process and thread management services
- Environment services
- Node internal communication and synchronization services
- Generalized input/output services
- File storage services
- Event, error, and exception management services
- Time services
- Memory management services
- Logical naming services

The services in this clause apply to a variety of architectures, including uniprocessor, multiprocessor, and distributed systems.

4.2.4.1.1 Process and Thread Management Services

These services relate to the creation, management, and deletion of entities executing within the scope of the application platform. Such entities are the means by which resources are managed. There is a broad spectrum of concurrency entities, ranging from the widely recognized operating system *process* to several variations on smaller executing entities, often called *threads* or *tasks*.

4.2.4.1.1.1 Processes

Processes are most often said to consist of an address space, a single (from a point of view outside of the process) flow of control that executes within that address space, and its associated system resources. At this end of the spectrum there tends to be substantial overhead for context (address space) switches, which are required to allow another process to make use of the processor. Each process is normally protected from others.

4.2.4.1.1.2 Threads

In contrast to processes, a thread or task is a concept of concurrency that takes place within the context of a single address space (process). Most often, many of the system resources allocated to a process are shared among all of the threads within that process, making threads typically “lighter weight” than processes, as substantial overhead is not required to change from one thread context to another. However, there are many other decisions that affect the “weight” of threads, thus creating the broad spectrum. The presence, absence, or nature of the following features are generally considered for these decisions:

- Identification
- Memory address space
- Memory allocation
- Stack size
- File/device status information
- Scheduling attributes
- Priority
- Interrupt handlers/interrupt vectors
- System resource limits
- Access control information

Threads are particularly useful for mapping asynchronous behavior into equivalent synchronous behavior, such as providing I/O parallelism or structuring applications composed of logically distinct tasks (e.g., simulations and windowing systems).

4.2.4.1.1.3 Process and Thread Services

It is possible for concurrency to take place at the process level, the thread level, or both levels simultaneously. Thus, *concurrency management* consists of those services that affect the execution of processes or threads, such as

- Stop and restart execution of processes or threads (suspend, resume)
- Modify processor allocation to a process or thread (priority, time slice)
- Modify scheduling of the process or thread based on timer (or other) events
- Protect the process or thread from interruption during critical periods
- Create a process or thread and make it ready for execution
- Destroy a process or thread and recover its resources
- Evaluate a reference to a process or thread
- Evaluate a connection to a process or thread, where a connection is a logical communication path between any two concurrency entities (process-process, process-thread, thread-thread)

Scheduling is the service that allocates processing resources to the processes or threads that are ready to execute. Typically, there are at least two types of scheduling occurring during execution: short term and long term. Long-term scheduling determines which possible requesters may request use of the processor resource at a given time; long-term scheduling may be incorporated into the design of the system. Short-term scheduling selects from among the active requesters that currently have need of the resource and allocates the processor resource to the selected requester.

The concurrency being managed may represent truly parallel execution in some systems. A request for a resource submitted at a local node need not always be serviced at that local node. The most advantageous way to service the request may result in part or all of the work being performed at a different processor node. Several reasons may cause this to occur, including load balancing, resource availability, computation speedup, hardware preference, and software preference. These services may hide from the application the fact that the functionality was being performed at a different node. This has the advantage that the code needs to know little about the system on which it is running. Alternatively, the services may allow the user to specify directly on which logical resource the function should be executed.

Priority scheduling of processor resources allows the request to be associated with its importance to use the service. More complex schemes also have a measure of the critical nature of the request that is used for graceful degradation purposes. The scheduler(s) use the priority and other information to arbitrate processor resource requests and to queue requests in the specific order.

Preemptive schedulers will deallocate a processor resource from a request when certain events occur. Usually, this is when a request of a higher priority or importance occurs or a specified time limit for the processor resource has expired.

Although threads are not synonymous with Ada *tasks*, Ada tasks are another example of a presumably lighter-weight unit of concurrency.

4.2.4.1.2 Environment Services

These services provide an application access to a variety of information relating to the environment in which the application is executing. The specific characteristics include the following:

- Attributes that are specific to concurrency entities (identification, priority, stack size, scheduling attributes, status, memory allocation)
- Processor-specific attributes (node identification, electronic nameplate information)
- User-specific attributes (user identification and terminal ID, user interaction profile, locale)
- Environment variables (command-line arguments, menu selections)
- Current time and date

4.2.4.1.3 Node Internal Communication and Synchronization Services

One or more applications and application subcomponents may run simultaneously on a processor within an application platform. The applications run as independent software entities and communicate among themselves via a variety of mechanisms provided or managed by the system services (see Figure 3.2). An important class of system services relates to the coordination and synchronization of these software entities. In traditional systems, entities execute on a single hardware processor. However, it is becoming common to have multiple processors and networked processors that place more requirements on the system services to provide coordination and synchronization among the many truly concurrent software entities.

When a platform has several application software entities executing concurrently, the applications need system services so that the entities can be coordinated and synchronized with each other. With respect to applications written using concurrency, there are two levels of concurrency that are usually seen by the application developer. The first level of concurrency, thread-level concurrency, is seen when the process executing the application software entity is split into multiple subcomponents (threads) that share access to the data and subprograms of the same application process. Concurrency services at this level concern the relative priorities and scheduling of threads within a single application program process and their communication with each other. At the second level of concurrency, application-process-level concurrency, a unit is a single application process including all its subcomponents. Concurrency services at this level concern the relative importance of the individual application processes competing for and sharing system resources.

These services are used to communicate among processes, among threads, and among processes and threads residing on the same node. The methods outlined do not include the network-specific services described in 4.3, but are limited to methods open to entities executing within the scope of a single application platform. Both synchronous and asynchronous services are defined. The specific services include the following:

- Create, delete, open, close, read, and write shared memory
- Create, delete, read, and write event flags
- Create, delete, set, and wait on semaphores
- Create, send, and receive signals
- Create, delete, open, close, send to, get from, and control message queues (IPC)
- Create, delete, send, and receive streams

4.2.4.1.4 Generalized Input/Output Services

These services are used by an application to perform generalized device I/O operations. (Frequently, I/O is provided through language services; however, the C language is an exception.) These services include synchronous and asynchronous operations for device- and class-specific functions: device initialization, device attachment,

asynchronous operation, and error notification. In addition, they may optionally include those services that are used to access specific device capabilities directly, particularly those services often referred to as *raw I/O*.

4.2.4.1.5 File Storage Services

Mass storage in the form of hierarchy of directories, subdirectories, and files may be available to an application executing within the application platform. The following subclauses describe the services available for creating, accessing, managing, and deleting these entities with mass storage. Both synchronous and asynchronous services are defined.

4.2.4.1.5.1 Naming and Directory Services

These services allow the access of files and directories through logical names rather than the actual hardware device naming conventions. The services allow sharing of files at various levels. For example, the services may not allow any shared naming of files and directories between systems, or they may allow shared files by explicit naming, or they may allow shared files by implicit naming. The directory services present a view or views of the directory to the application or end user.

4.2.4.1.5.2 File Modification Primitives

Primitive services for files and directories are

- Read a portion of the file
- Write to a portion of the file
- Open access to a file
- Create a new file
- Close access to a file
- Delete a file
- Support read and write locks at both the record and file levels

These services may be very complex. For example, the access to read or write may be direct (by record number), sequential (one record at a time), or indexed (by a key). The services also may support a variety of file structures, including linked, segmented, contiguous, serial, and directory.

4.2.4.1.5.3 File Support Services

Additional services support the physical devices on which the files and directory reside. These services include the mounting and dismounting of media, the formatting of media, and the partitioning of media.

File optimizations for realtime services may provide advisory information on the behavior of the application with respect to the data in a file. This information may be used to optimize the handling of specified data and may affect the performance of other operations without having any effect on the semantics of other operations.

4.2.4.1.6 Event, Error, and Exception Management Services

These services provide a common facility for the generation and communication of asynchronous events among the application platform and application programs. A major use of the event services is to report error conditions, but they are also used to provide an indication of some condition to the application programs. These services include the following:

- Event and error receipt
- Event and error distribution
- Event and error management, including user-selectable error processing alternatives (filtering, retry, ignore, accumulate occurrences)

- Event logging
- Enable and disable application-controlled interrupts
- Mask and unmask application-controlled interrupts

4.2.4.1.7 Time and Timer Services

Timers may be a static or dynamic resource on the system, necessitating a variety of allocation and management strategies. These services are used by applications to perform a variety of services based on absolute and relative time. These services include the following:

- Create a timer
- Delete a timer
- Initiate the measurement of an arbitrarily specified time duration
- Set the timer to generate an application-specified realtime signal
- Read the current value of a timer
- Simultaneously initialize a timer with a value pair (initial value and reload value), and arm the timer
- Simultaneously initialize a timer with a value pair, and trigger the timer

4.2.4.1.8 Memory Management Services

These services are used by application processes and threads to manage their memory space. The service of allocating additional memory to the process and returning it to the system is provided in most programming languages and, consequently, is not included in this clause. Additional capabilities to provide system services to manage application needs for virtual and fixed memory are provided. Such needs include memory-mapped files and shared memory. Memory-mapped files provide a mechanism that allows processes and threads to access files by directly incorporating file data into the process address space; this can significantly reduce I/O data movement. Shared memory spaces are named regions of storage that can be mapped into the address space of one or more processes to allow them to share the associated memory. There is also a service for locking pages into physical memory to support applications that need to bypass the abstractions provided by the virtual memory system.

4.2.4.1.9 Logical Naming Services

These services allow the usage of system resources through logical names rather than the actual hardware device naming conventions. Furthermore, they allow the resources of other processor nodes to be accessed via a logical name so that no knowledge of the location of the resource is needed and also so that the location may change over time. Logical names are also used by security services to hide resources from unauthorized processes by only letting authorized processes know the logical name that is needed to use the physical resource.

The logical name to physical name relationship can be one-to-many, many-to-one, or many-to-many. Frequently, one physical resource may have multiple logical names; likewise, one logical name may represent a “bank” of available physical resources. These services have to provide the proper resolution of names, logical and physical, in all of these cases.

4.2.4.2 EEI Services

There are no unique core system services EEIs. There are many important standards, such as backplane or peripheral attachment specifications, describing interfaces that could be considered to be part of such an EEI. Examples of these specifications might include SCSI, MULTIBUS, and others. However, from the perspective of this guide, unless these specifications are used for platform interoperability, they are not considered to be part of the EEI. These specifications are PIIs, as described in 3.6.

4.2.5 Standards, Specifications, and Gaps

See Table 4.3.

Table 4.3—Core System Services Standards

Service	Type	Specification	Subclause
Process management	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.2d-1994 {118}	4.2.5.1
Thread management	S	IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Environment	S	ISO/IEC 9945-1 :1990 {68}	4.2.5.1
Node internal communication/ synchronization	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Generalized I/O	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
File storage	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Event, error, and exception	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Time and timer	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Memory management	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}, IEEE Std 1003.1c-1995 {117}	4.2.5.1
Logical naming	S	ISO/IEC 9945-1 :1990 {68}	4.2.5.1
Scheduling	S	ISO/IEC 9945-1 :1990 {68}, IEEE Std 1003.1b-1993 {116}	4.2.5.1
Asynchronous I/O	S	IEEE Std 1003.1b-1993 {116}	4.2.5.1
Synchronous I/O	S	IEEE Std 1003.1b-1993 {116}	4.2.5.1
Realtime signals (or asynchronous event notification)	S	IEEE Std 1003.1b-1993 {116}	4.2.5.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.4 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

Table 4.4—Systems Services Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
ISO/IEC 9945-1 :1990 {68}	E	S			S		
IEEE Std 1003.1b-1993 {116}		E			S		
IEEE Std 1003.1c-1995 {117}		E			S		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
ISO/IEC 9945-1 :1990 {68}		S				
IEEE1003.1b-1993 {116}		E				
IEEE1003.1c-1995 {117}		E				

NOTES:

1 — LIS — Language-independent specification is available.

2 — Ada, APL, Basic, ... — Language-dependent specification is available.

3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.2.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.2.5.1.1 ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990) {68}

The purpose of ISO/IEC 9945-1 :1990 {68} is to define a standard operating system interface based on the UNIX® operating system documentation to support application portability at the source-code level. The standard is intended for systems implementors and applications software developers.

In addition to ISO/IEC 9945-1 :1990 {68}, another standard has been published on test methods for verification of POSIX standards: ISO/IEC 13210 :1994 {97}, which is identical to IEEE Std 1003.3-1991.

Table 4.5 outlines the contents of ISO/IEC 9945-1 {68}. Revisions are currently in progress to deal with

- A language-independent services specification
- A unified data interchange format
- Miscellaneous functions identified by comments on the current standard

ISO/IEC 9945-1 :1990 {68} draws heavily upon major implementations of the UNIX operating system, including the System V and BSD versions. Where a specific behavior was clearly needed (e.g., signals), only a single behavior was permitted. However, there are points where functions were considered optional and others where two different behaviors were considered acceptable. However, in many cases, a solid technical argument favoring one approach over the other was not established. In this case, two behaviors (usually System V and BSD) were defined as being permitted. This is beneficial for writing portable applications, since those that can tolerate both behaviors will run on a wider range of systems. This also causes a slight disadvantage in writing such applications, since it can mean having to manage more variations in behavior.

Table 4.5—Functionality of ISO/IEC 9945-1 :1990

File system organization and file naming conventions
System configuration and file system configuration characteristics
Error messages and reporting mechanism (<i>errno</i>)
Application environment information (<i>environ</i>)
Process creation, management, and termination: <i>exec</i> (), <i>fork</i> (), <i>wait</i> ()
Process environment: user ID, process ID, group ID
Exception conditions and handling (signals)
Timer operations
File and directory operations: FIFO files, pipes, status, open/close, read/write
File protection mechanisms
Record and file locking mechanism
Device specific functions: terminal controls: processing modes: echo, baud rate, modem termination
C-language specific routines: <i>setlocale</i> (), nonlocal jumps
User and group database information (excluding password information)
Data interchange formats (USTAR and CPIO)
Also included is an informative annex that provides insight on the selection of various functions and features, including some guidance to developers to understand what types of variations may exist and how that can impact portability.

There is a long-range plan for the evolution of ISO/IEC 9945-1 :1990 {68} into a more complete standard. Under this plan, as related standards such as IEEE Std 1003.1b-1993 {116}, IEEE P1003.1e {B21} (see 5.2.5.2.1), and IEEE P1003.1f {B22} (see 4.3.5.2.1) are approved as international amendments, they will be incorporated into ISO/IEC 9945-1 :1990 {68}. IEEE Std 1003.1b-1993 {116} has already been incorporated based on this plan and a merged document is now available from the IEEE.

NOTE — FIPS Publication 151-2 {B18} is a profile of the base standard ISO/IEC 9945-1 :1990 {68}.

4.2.5.1.2 IEEE Std 1003.1b-1993

Interfaces to realtime extensions to ISO/IEC 9945-1 :1990 {68} are defined in IEEE Std 1003.1b-1993 {116}. This amendment provides standardized interfaces to the following functions:

- Response to asynchronous events
- Priority interrupts and scheduling
- Preemptive scheduling
- Memory locking
- Realtime timers (with nanosecond resolution times)
- Shared memory
- Semaphores
- Message queues
- Asynchronous event notification
- Asynchronous I/O
- Synchronized I/O

4.2.5.1.3 IEEE Std 1003.1c-1995

Interfaces to “pthreads,” the POSIX instantiation of the threads concept (see 4.2.4.1.1) are defined in IEEE Std 1003.1c-1995 {117}.

4.2.5.2 Additional Specifications

None.

4.2.5.3 Unaddressed Services

There is a significant aspect of the core system services for which no standards or public specifications currently exist: the considerations implied by the use of multiprocessors and distributed systems to implement some or all of the core system services.

ISO/IEC9945-1 :1990 {68} and IEEE Std 1003.1b-1993 {116} do not yet address all of the services indicated in 4.2.4. Areas of shortfall include event, error, and exception management services, high-performance (or realtime) file systems, and some generalized I/O services. In addition, security (see 5.2), reliability, availability, and maintainability services are not reflected in these two base standards, and some capabilities are explicitly considered to be implementation defined. Finally, since these are base standards (or, in the case of IEEE Std 1003.1b-1993 {116}, an extension to a base standard), profiles are needed in order to select appropriate features and provide appropriate combinations with other related capabilities.

X/Open XPG4 {B69} and OSF AES-OSC {B50} provide specifications for some of these unaddressed services.

4.2.6 POSIX OSE Cross-Category Services

4.2.6.1 Security Services

The security services related to the core systems services include the following:

- Prevention of unauthorized access
- Prevention of data compromise
- Prevention of service denial
- Security administration

The security services allow the management of the security system, including the administration of permissions to personnel, data, and services as well as capability lists. In addition, they permit the administration of access mechanisms (most often passwords and capability lists) and services that allow the system to switch modes of operation. The services will likely be accessed by the target system operator with security responsibilities through the target system operator services.

4.2.6.2 System Management Services

The system management services related to the core systems services include the following:

- Resource management services
- System operator services
- System administration

4.2.7 Related Standards

The following standards and emerging standards are related to the services covered in this clause, inasmuch as they address at some level services either explicitly listed in, or implied by, the services found in 4.2.4:

IEEE P1003.1e {B21}Security interface for POSIX
ISO/IEC 9945-2 :1993 {69}Shell and utilities (including the User Portability Option in Section 5)
IEEE Std 1003.2d-1994 {118}Batch environment
IEEE P1387 System administration

The Ada language provides some of the core system services. While there are no explicit APIs for Ada runtime features (other than the language itself), language runtime services need to be effectively integrated with the core system services in order for applications to work. Ada is discussed in 4.1.

4.3 Communication Services

4.3.1 Overview and Rationale

This clause describes the communication services component of the application platform. It also describes the services provided to application programs and users, and it describes current and emerging standards that are addressing these services.

Applications gain direct access to communication services via standardized APIs. However, many of the services are expressed in terms of “network” capabilities and are traditionally thought of as network services. Nonetheless, they can be used as communication services without ever interacting across a physical network.

4.3.2 Scope

Communication services support distributed environments, such as file transfer, name space and directory services, electronic mail, virtual terminal services, transparent file access, process-to-process communication, remote procedure call, data representation services, and distributed time management. The application programs using these services should be able to access them via a high-level, context-insensitive or low-level, context-dependent interface.

The network protocols defined for both Open Systems Interconnect (OSI) and Internet Protocol Suite (IPS; i.e., TCP/IP) should provide the basis for the open networking interfaces; however, these interfaces should not preclude the use of some subsequent networking protocol in the future. The interfaces provided by the network services have to be network protocol-independent and provide for this level of interoperability.

4.3.3 Reference Model

This subclause identifies the entities and interfaces specific to the construction of a POSIX communication environment. This environment is consistent with and extends the environment of Section 3.

As illustrated in Figure 4.3, the components of a communication architecture that require standardization are divided into an EEI and API.

4.3.4 Services

The services for both the API and EEI are described in this subclause.

4.3.4.1 API Services

This subclause describes the major categories of communication services available at the communication services API. These services are

- Transparent network services
- Directory services
- Application-to-Platform services
- Application-to-Application communication services
- Distributed system services

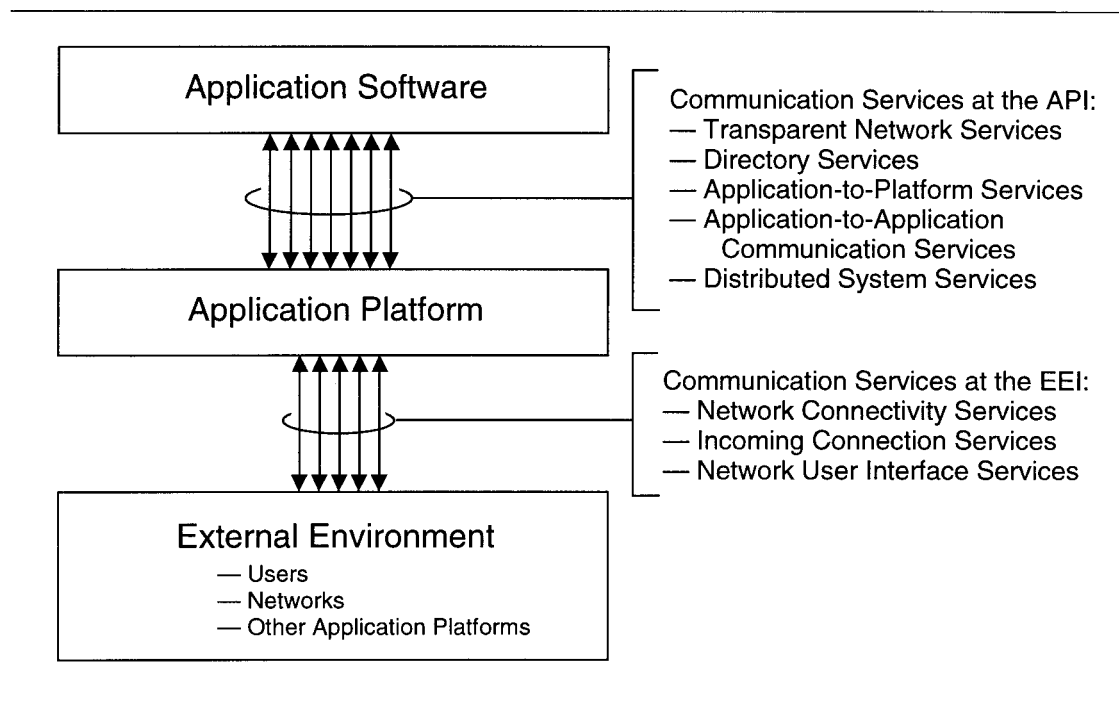


Figure 4.3—POSIX OSE Communication Services Reference Model

4.3.4.1.1 Transparent Network Services

Transparent network services are those services typically provided by the core system services (such as file access), but that can be extended across a network with no modifications to the application. An example of this might be a networked file system. When one of these core system services is extended transparently across a network, some additional capabilities may need to be made available to the application. These include extended error conditions or the ability to query whether or not a particular system service is available on a particular file. The extensions to these services to extend them transparently across a network are described in this clause. These services include the ability to

- Query which services are supported on particular files or directories
- Declare the intent to accept less than full file semantics for those files accessed across a network

4.3.4.1.2 Directory Services

Directory services allow an application to find the names and addresses of objects and services available to the application. These services include the ability to

- Read a named directory entry
- Compare an attribute value with the value stored in the directory
- List the immediate subordinates of a directory entry
- Search a portion of the directory information tree for specific entries
- Abandon the result of pending asynchronous operation
- Add, modify, or remove a leaf entry from the directory information tree
- Modify the name of a leaf entry
- Specify and enforce access controls for directory entries

4.3.4.1.3 Application-to-Platform Communication Services

The application-to-platform communication services are the services requested by the application that are performed by the application platform on behalf of the application, without the application actually communicating directly with another application. Many of these services may actually connect to some remote application, but the details of the connection are left up to the application platform.

These services will be provided by a relatively simple high-level API. These services include

- File transfer
- Electronic mail
- Virtual terminal

4.3.4.1.4 Application-to-Application Communication Services

There are three areas of application-to-application communication services:

- Remote Procedure Call (RPC) Services
- Simple Communication Services
- Detailed Communication Services

These communications services can be used to allow applications to communicate across a physical network. In addition, these services can be used to allow applications on the same application platform to communicate. Many of these services can be used transparently, either for local communication or communication across a network.

The RPC services allow an application to register with the application platform as the provider for a particular RPC service. Once the service has been properly registered, other applications can transparently request services using a subroutine call. The details of communicating the service request to the application that is registered to provide the service and the return of the response to the requesting application are handled transparently by the application platform.

Applications making use of RPC services need not be aware that the services are being provided via an RPC mechanism.

The simple communication services are application-to-application services provided using a simple set of interface routines. These will allow a wide variety of applications to be written that do not need to exercise control over their communications at a very complex level of detail.

In addition, these services should be provided over a wide variety of communication transport mechanisms. Applications written exclusively using the simple communication services should be portable across a wide variety of network environments.

Applications written using the simple communication services need not be able to make use of unique advantages of a particular physical networking scheme. To make full use of all features of a particular underlying network implementation, detailed communication services need to be used.

The services for the simple communication services are intended to be the minimum necessary to write a large subset of communications applications.

The simple communication services give up the capability to control every detail of the communication services in the interest of portability across communication environments and applications simplicity.

The detailed communication services API allows the application to exercise control over specific features supported by the underlying communication implementation. In addition, an application using the detailed communication services may be able to make use of unique communication capabilities available in particular communication environments.

4.3.4.1.4.1 RPC Services

These services include the ability

- To define the RPC interfaces using an interface definition language
- To transfer bulk data between applications efficiently
- To make use of directory services via RPC
- To make use of security services
- To interact with thread services
- To register as an RPC service provider
- To wait for incoming requests
- To control parameters such as timeout for an application using RPC services

4.3.4.1.4.2 Simple Communication Services

The services provided at the simple communication interface are

- 1) Connection-oriented services
 - a) Indicate willingness to accept incoming connections
 - b) Establish and destroy connections
 - c) Transfer data over connections
 - Read
 - Read with timeout
 - Write
 - Write with timeout
 - Nonblocking I/O
 - d) Handle I/O events in a simple manner
 - e) Handle errors in a simple manner
 - Connection dropped notification
 - Connection read failure
 - Connection write failure
 - f) Close a connection
 - Unconditionally
 - Only after all data has been received
- 2) Connectionless services
 - a) Indicate willingness to accept incoming requests
 - b) Send requests
 - Without acknowledgment
 - Specified timeout
 - c) Receive requests
 - Wait unconditionally
 - Wait with timeout
 - d) Query whether any requests are available
 - e) Handle event notification in a simple manner
 - Lost request
 - Request acknowledgment
 - f) Handle errors in a simple manner
 - General network failure
- 3) Support for server applications
 - The ability to register as the provider for a service

- 4) Simple status inquiry
 - General network availability

4.3.4.1.4.3 Detailed Communication Services

The services provided to the application at the detailed communication interface include all of the services in the simple communication services plus the following abilities:

- Query the network services to get detailed information about network configuration and status
- Specify performance metrics
- Control routing
- Select between different network protocols
- Negotiate capabilities
 - Required capabilities
 - Optional capabilities
 - Determine the results of the negotiation
- Request information with different priorities
- Request and process extended event notification
- Request and process extended error recovery, including allowing the application to control error recovery completely
- Make full use of network resources for performance-critical applications

This should provide the application with the ability to control connection-oriented services and connectionless services completely.

4.3.4.1.5 Distributed System Services

The services provided in this area include the ability to

- Identify available resources in a distributed system
- Make use of the resources in a distributed system dynamically
- Access files, regardless of the physical location of the files
- Have reliable time services across all of the resources of the distributed system

4.3.4.2 EEI Services

4.3.4.2.1 Network User Interface Services

Services provided at the EEI in the area of user interfaces to network tools include

- Transfer and access files between application platforms, including appropriate conversion as necessary
- Execute commands on remote application platforms
- Login to remote application platforms
- Mail messages to users on the local and remote application platforms
- Access directory information and local remote application platforms

These networking commands should provide appropriate security capabilities to allow each application platform to configure the level of security required for incoming requests.

4.3.4.2.2 Network Connectivity Services

Services provided at the EEI in the area of network connectivity include the ability to

- Connect and interoperate with other standards-based protocols, such as the OSI protocol suite
- Connect and interoperate with systems using de facto networking standards, such as TCP/IP

- Connect and interoperate with personal computer and workstation networks

4.3.4.2.3 Incoming Connection Services

These EEI services are those provided to incoming connections that are not directed to an end-user application or server. These incoming connections are directed to standard services that can be provided by the application platform. These services include

- Virtual terminal
- Remote execution of commands
- File transfer services
- Remote authentication
- Remote data access
- Remote status information
- Mail delivery services
- Directory services

To access these services, each user need not provide an application on the remote host. Simply by connecting to the service, the application platform will provide the service.

4.3.5 Standards, Specifications, and Gaps

Table 4.6 lists API standards.

Table 4.6—Communication Standards — APIs

Service	Type	Specification	Subclause
Transparent network	E	IEEE P1003.1f (B22) (TFA API)	4.3.5.2.1
Directory	S	IEEE Std 1224.2-1993 {125}, IEEE Std 1327.2-1993 {130}	4.3.5.1
Application-to-System	S	IEEE Std 1224.1-1993 {124}, IEEE Std 1327.1-1993 {129}	4.3.5.1
RPC	P	OSF DCE RPC {B51}	4.3.5.2.2
Simple communication	E	IEEE P1003.1g {B23}	4.3.5.2.1
Detailed communication			
Application layer	S	IEEE Std 1238.1-1994 {126} (ACSE API)	4.3.5.1
Transport layer	E	IEEE P1003.1g {B23} (Transport API)	4.3.5.2.1
Distributed system	S	ISO/IEC 10026-1 : 1992 {72}	4.3.5.1
	E	IEEE P1003.1f {B22} (TFA API)	4.3.5.2.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.7 lists EEI standards. The entries are intended to be only a representative sample of the many available networking EEI standards. There are a number of publications that provide a detailed list of communications standards and a large number of ISO profiles that explain and show how to use these communications standards in various combinations for different networks and purposes.

To make a useful networking interface that can be implemented on a wide variety of computing systems, multiple network standards need to be combined into a single unified networking profile.

Table 4.8 lists standards for services provided by the application platform at the EEI.

Table 4.9 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

Table 4.7—Communication Standards — EEIs

Service	Type	Specification	Subclause
Standardized Profiles	P	FIPS Publication 146-1 {B17}, UK GOSIP {B47}	4.3.5.2.2
Network connectivity			
OSI connectivity			
General			
Data representation	S	ISO/IEC 8824 : 1990 {41}, ISO/IEC 8825 : 1990 {42} (ASN.1)	4.3.5.1
Connection-Oriented			
Application layer	S	ISO8649 :1988 {30}, ISO 8650 : 1988 {31}	4.3.5.1
Presentation layer	S	ISO/IEC 8823-1 :1994 {40}	4.3.5.1
Session layer	S	ISO8327 :1987 {23}	4.3.5.1
Transport layer	S	ISO/IEC 8073 : 1992 {22}	4.3.5.1
Connectionless			
Application layer	S	ISO 8649 : 1988 {30} Amd. 2	4.3.5.1
Presentation layer	S	ISO/IEC 9576 : 1991 {54}	4.3.5.1
Session layer	S	ISO/IEC DIS 9548 {53}	4.3.5.1
Transport layer	S	ISO/IEC 8072 :1994 {21} ISO 8602 :1987 {27}	4.3.5.1
TCP/IP connectivity			
Transport protocol	P	RFC-793 {B37} TCP, RFC-791 {36} IP	4.3.5.2.2
Physical connectivity	S	ITU-T X.25 {101}, ISO/IEC 8802-3 :1993 {34} (Ethernet), ISO/IEC 8802-4 :1990 {35} (Token Bus), ISO/IEC 8802-5 :1992 {36} (Token Ring), ITU-T I.120 {99} (ISDN)	4.3.5.1
Distributed data	S	ISO/IEC 9314 {50} (FDDI), ISO 9316 :1989 {50} (SCSI), ANSI X3.131-1994 {107} (SCSI-2)	4.3.5.1
NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap			

4.3.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.3.5.1.1 API Standards

IEEE Std 1224-1993 {123}
 IEEE Std 1224.1-1993 {124}
 IEEE Std 1327-1993 {128}
 IEEE Std 1327.1-1993 {129}

These standards provide an X.400 API and a companion OSI object management API, based on the X.400 API and an OSI Management API developed by the X.400 API Association and X/Open. The X.400 API consists of two parts: an X.400 application API and an X.400 gateway API. This APIs were developed based on the 1988 ITUT-T X.400 {102} series of recommendations. The X.400 API and object management API are separate documents. Each of the API standards (IEEE Std 1224-1993 {123} and IEEE Std 1224.1-1993 {124}) are language-independent specifications (LIS). The corresponding C-language bindings are IEEE Std 1327-1993 {128} and IEEE Std 1327.1-1993 {129}.

The purpose of the X.400 API is to provide standard interfaces supporting the development of applications that use the message transfer system and gateways that incorporate or use X.400 mail functionality; this includes gateways between X.400 mail networks and proprietary mail systems.

The purpose of the companion OSI object management API is to provide a standard interface supporting the manipulation of complex arguments and parameters used by the X.400 and directory services APIs. The scope of the OSI object management API is to define an ASN.1 object management API for use in conjunction with, but otherwise independent of, the X.400 and directory services APIs that are currently being standardized.

IEEE Std 1224.2-1993 {125}
 IEEE Std 1327.2-1993 {130}

These API standards enable applications to access directory services. IEEE Std 1327.2-1993 {130} is the C-language binding to the language-independent IEEE Std 1224.2-1993 {125} (which was originally called IEEE P1003.17). The directory services available through the API are based closely on the abstract services specified in the ISO/IEC 9594-1 : 1990 {58} (X.500) directory standards. However, the API may be used to access the following directory services:

- X.500
- Domain Name System
- Others

Table 4.8—Communication Standards — Services at the EEI

Service	Type	Specification	Subclause
Directory	S	ISO/IEC 9594-1 : 1990 {58} (X.500)	4.3.5.1
	P	RFC-1034 {B43} (Domain naming)	4.3.5.2.2
Message handling	S	ITU-T X.400 {102}	4.3.5.1
	P	RFC-821 {B38}, RFC-822 {B39} (SMTP)	4.3.5.2.2
File transfer	S	ISO 8571 {25}, ISO/IEC ISP 10607 {86}	4.3.5.1
	P	RFC-959 {B41} (FTP), RFC-1014 {B42} RFC-1050 {B44}, RFC-1094 {B45} (NFS)	4.3.5.2.2
Virtual terminal	S	ISO 9040 : 1990 {45}, ISO 9041 {46}	4.3.5.1
	P	RFC-854 {B40} (Telnet)	4.3.5.2.2

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.9—Communication Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
IEEE P1003.1f {B22}	E				E		
IEEE P1003.1g {B23}	E				E		
IEEE Std 1224-1993 {123}	S				S		
IEEE Std 1224.1-1993 {124} (X.400)	S				S		
IEEE Std 1224.2-1993 {125} (X.500)	S				S		
IEEE Std 1238.1-1994 {126} (ASCE)	E				E		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
IEEE P1003.1f {B22}						
IEEE P1003.1g {B23}						
IEEE Std 1224-1993 {123}						
IEEE Std 1224.1-1993 {124} (X.400)						
IEEE Std 1224.2-1993 {125} (X.500)						
IEEE Std 1238.1-1994 {126} (ASCE)						

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.3.5.1.2 EEI Standards**4.3.5.1.2.1 Connection Oriented Application-to-Application Communication Services**

ISO 8649 :1988 {30}

ISO 8650 :1988 {31}

These standards describe the services provided and protocol used by the connection-oriented Application Layer Association Control Service Element.

ISO/IEC 8823-1 :1994 {40}

This standard describes the connection-oriented presentation layer services and protocols.

ISO 8327 :1987 {23}

This standard describes the connection-oriented session layer services, protocols, and formats.

ISO/IEC 8073 :1992 {22}

This standard describes the connection-oriented transport layer services and protocols.

4.3.5.1.2.2 Connectionless Application-to-Application Communication Services

ISO 8649 :1988 {30} Amd. 2

This standard describes the services provided and protocol used by the connectionless Application Layer Association Control Service Element.

ISO/IEC 9576 :1991 {54}

This standard describes the connectionless presentation layer protocol specification.

ISO/IEC DIS 9548 {53}

This draft standard describes the connectionless session services.

ISO/IEC 8072 :1994 {21}

This standard describes the connectionless mode transport services.

ISO 8602 :1987 {27}

This standard describes how a connectionless transport can utilize either a connectionless network layer or a connection-oriented network.

4.3.5.1.2.3 Physical Connectivity Standards

ITU-T X.25 {101}

This is a series of standards for packet-switched data networks.

ISO/IEC 8802-3 :1993 {34}

This standard is based on the 10 Mb/s Ethernet local-area network (LAN) technology.

ISO/IEC 8802-4 :1990 {35}

This standard specifies how to connect to a token bus LAN.

ISO/IEC 8802-5 :1992 {36}

This standard specifies how to connect to a token ring LAN.

ITU-T I Series

These series of standards pertain to ISDN networking:

I.100 series General information

I.200 series ISDN services

I.300 series Network internals

I.400 series User-to-network interface

I.500 series Interfacing ISDN to other networks

I.600 series Maintenance

ISO/IEC 9314 {50}

This standard specifies aspects of FDDI.

ISO 9316 : 1989 {51}

This standard specifies the Small Computer System Interface (SCSI).

ANSI X3.131-1994 {107}

This standard specifies SCSI-2 (an extension and revision of SCSI).

4.3.5.1.2.4 General Communication Standards

ISO/IEC 8824 :1990 {41}

ISO/IEC 8824 Amd. 1 {41}

ISO/IEC 8825 :1990 {42}

These standards describe Abstract Syntax Notation One (ASN.1).

ISO/IEC 9594 {58}

This is a multipart standard that describes aspects of directory services.

ITU-T X.400 {102}

This series of standards specifies mail header format and mail transfer protocol. There is also an international standard for portions of X.400 (ISO/IEC10021 {71}).

ISO 8571 {25} This series of standards specifies the application layer entity for FTAM—File Transfer, Access, and Management.

ISO/IEC ISP 10607 {86}

This is an ISP for FTAM that selects various FTAM options.

IEEE Std 1238.1-1994 {126}

This is an API standard for interfacing with the FTAM application layer element.

ISO/IEC 9596-1 :1991 {60}

This standard deals with the exchange of information related to the management of the communication aspects of open systems.

4.3.5.2 Additional Specifications

4.3.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

Figure 4.4 shows the logical relationship between the various POSIX networking API standards. However, the figure does not necessarily reflect the way that those APIs are to be implemented. There are many possible implementations of these APIs. See 3.6 for more information on possible implementations of the application platform.

4.3.5.2.1.1 IEEE P1003.8

The IEEE P1003.8 working group is developing an amendment to ISO/IEC 9945-1 :1990 {68} (IEEE P1003.1f {B22}) to extend and circumscribe the file aspects of the system API to support file access mechanisms incapable of supporting the full behavior required or that provide access to files via a networked mechanism.

The draft amendment defines additional semantics for those base services where specific errors may be returned due to the nature of the implementation used to provide access to a file. In addition, the amendment specifies a mechanism that an application can use to determine which services can be used on a particular file.

4.3.5.2.1.2 IEEE P1003.12

The IEEE P1003.12 working group is developing a standard that provides networking APIs. IEEE P1003.1g {B23} contains the specification for a Simple Network Interface (SNI) and a Detailed Network Interface (DNI). The SNI is designed to be used over a number of different transport services, ranging from ISO networks to completely proprietary networks, without requiring application changes. To do this, the SNI has a very limited set of services with minimal parameters.

The DNI is also designed to be implemented across a wide variety of network protocols. However, DNI allows applications to access the low-level details of each of the different network protocols. As a result, programs written using DNI may be portable between environments that use the same underlying network protocols.

Applications can be written using a combination of the SNI and DNI interfaces. The engineers designing the applications can make portability tradeoffs as the applications are developed.

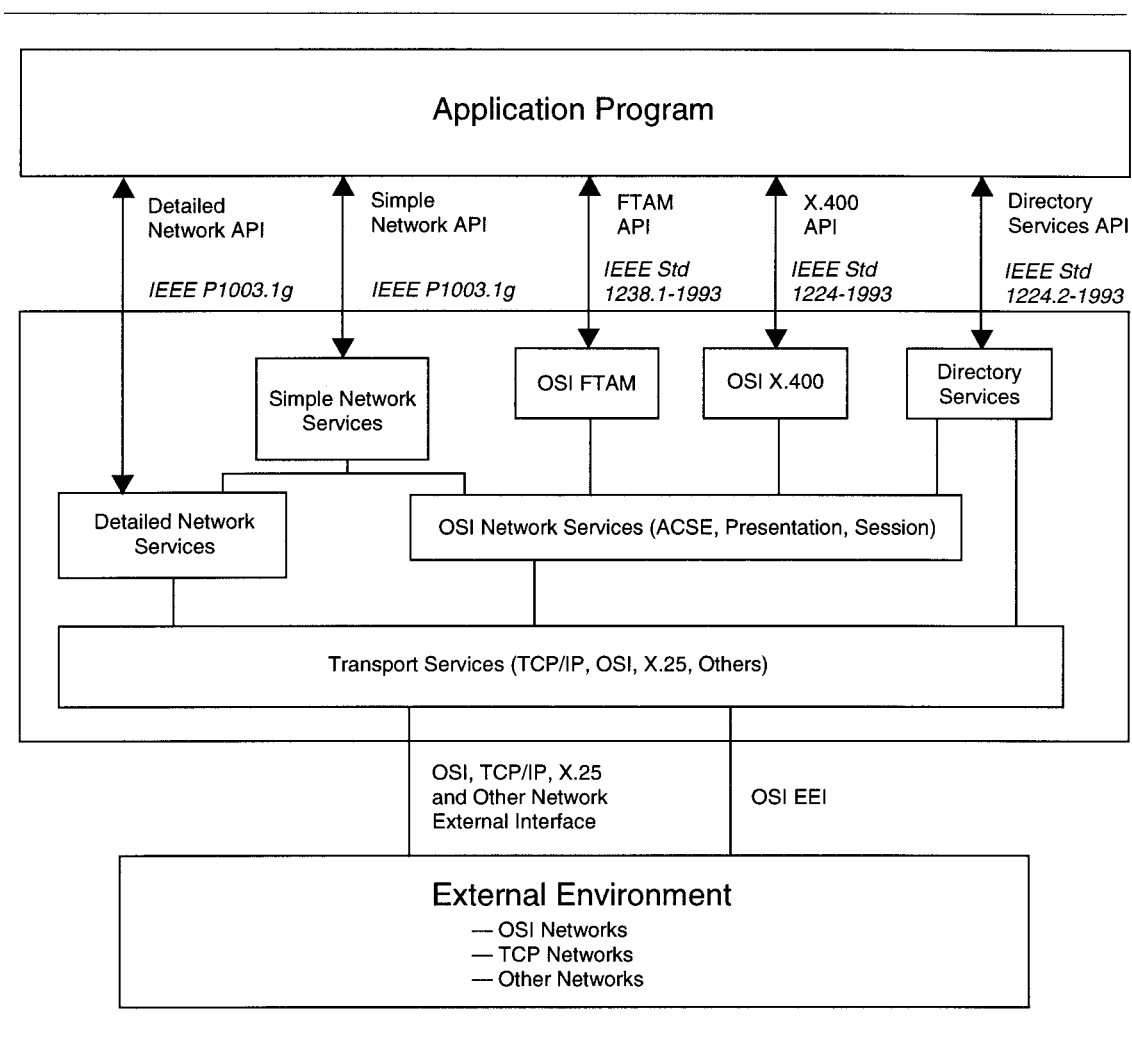


Figure 4.4—POSIX OSE Communication Standards

4.3.5.2.1.3 Distributed Data Technology

This is an area of technology that includes work to extend and revise the Fibre Distributed Data Interface (FDDI) and the Small Computer System Interface (SCSI). This work is being produced by ASCs X3T10, X3T11, and X3T12, along with corresponding projects in ISO/IEC JTC 1.

4.3.5.2.2 Public Specifications

The specifications listed in this subclause are not part of the POSIX OSE, but may be of interest. Use of these specifications should be considered carefully because there are risks in using these unapproved specifications. They are included in this guide to indicate some of the existing work that has been done in areas that are gaps in the POSIX OSE.

4.3.5.2.2.1 Government OSI Profiles

National and regional governments are creating profiles that specify the set of networking standards and options to be used in government procurements. These profiles may be mandatory in certain areas. Examples of such profiles are US GOSIP V2.0 {B17} and UK GOSIP {B47}.

4.3.5.2.2.2 IPS (TCP/IP) Networking

The Internet Protocol Suite (IPS) is a set of standards that effectively makes up a profile. This profile is commonly referred to as TCP/IP networking. The common standards that make up the IPS are as follows:

RFC-791 {B36}	
RFC-793 {B37}	Transport Protocol
RFC-959 {B41}	File Transfer Protocol
RFC-821 {B38}	
RFC-822 {B39}	Simple Mail Transfer Protocol
RFC-854 {B40}	Virtual Terminal Service
RFC-1034 {B43}	Domain Name Service

The specifications for these network protocols are currently in the form of Request For Comments (RFCs), under the control of the Internet Activities Board (IAB). Although not part of the formal standards process, the IPS is the de facto standard for internetworking in noncommercial, public networks and many private internetworks. Commercial implementations are available for almost all platforms, and free implementations are available for most desktop platforms.

4.3.5.2.2.3 OSF DCE RPC

The OSF DCE RPC {B51} provides remote procedure call services as described in 4.3.4.1.4.1.

4.3.5.2.2.4 NFS

Network File System (NFS) is a widely used specification for file sharing protocols. NFS is described in the following documents:

RFC-1094 {B45}	NFS: Network File System Protocol Specification
RFC-1014 {B42}	XDR: External Data Representation Standard
RFC-1050 {B44}	RPC: Remote Procedure Call Protocol Specification

While the XDR and RPC specifications that are part of NFS are written as general-purpose specifications, their primary use is in the context of NFS.

4.3.5.3 Unaddressed Services

The areas in which there are unaddressed services include

- Distributed system services
- Distributed security APIs and protocols
- Distributed systems management
- Standards for interoperability (APIs and protocols) between heterogeneous systems (from PCs to mainframes)
- User interfaces to network services

X/Open XPG4 {B69} and OSF AES-OSC {B50} provide specifications for some of these unaddressed services.

4.3.6 POSIX OSE Cross-Category Services

4.3.6.1 Network Management

The POSIX OSE cross-category services in the area of network management include

- Manage
 - Network objects
 - Network relationships
 - Network security
- Monitor and report on
 - Network events
 - Network service alarms
 - Network security alarms
- Log
 - Network events
 - Network availability
 - Network load
 - Network performance
- Test network performance and reliability
- Query the status of other application platforms on the network

4.3.7 Related Standards

ISO/IEC 10026-1 :1992 {72}, Parts 1-3.

4.4 Database Services

4.4.1 Overview and Rationale

This clause gives an overview of an architectural framework for discussing database management, including the services provided to application programs and users. It also describes current and emerging standards for those database services.

Database management is an important component of the POSIX OSE; database access through a database management system plays a key role in a large class of application programs, especially those used in business. For portability and interoperability, an application using a database has to be isolated from the hardware and software retrieval methods as much as possible. Where performance is the key issue, the data manipulation capability might be coded into the application programs, but this is done at the cost of interoperability and portability.

4.4.2 Scope

This subclause discusses the scope of services provided by the database management component to both application and utility programs. Services offered by this component have been selected from those normally provided by indexed files and databases based on hierarchical, network, relational, and object-oriented “data models.” Services that are not normally provided as part of a database have been excluded. In systems where applications share large amounts of data, in cases where coordination with multiple, heterogeneous resource managers is required, or to reduce response times, transaction processing services may also be needed. See 4.6.

4.4.3 Reference Model

In this subclause, the conventional view of database management is related to the POSIX OSE reference model described in 3.2.

The conventional view of the database model is introduced first. A database application program, through a *database services API*, requests database services. For convenience in the following discussion, the agent responsible for providing those services is called the *database manager*. The database manager is responsible for providing the application access to the *database*. See Figure 4.5.

The conventional view in Figure 4.5 also covers the concept of a *database utility program*: one or more special application programs, usually provided by a database vendor, that perform utility services on the database. Such utilities might reorganize the database, recover the database after a system failure, etc.

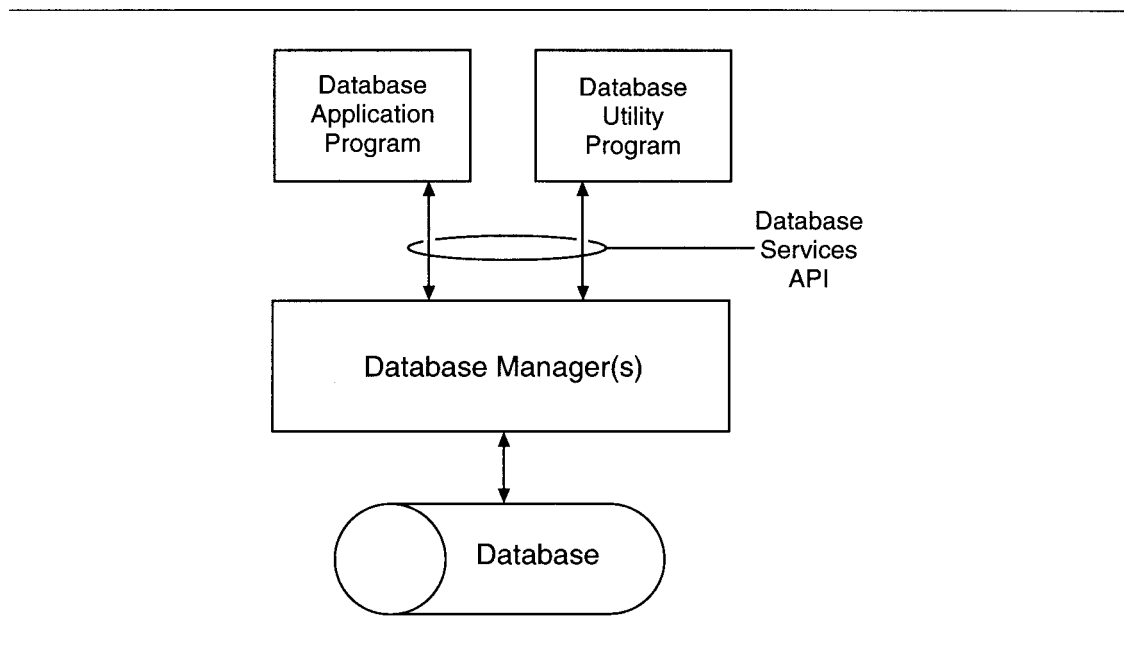


Figure 4.5—Traditional Database Model

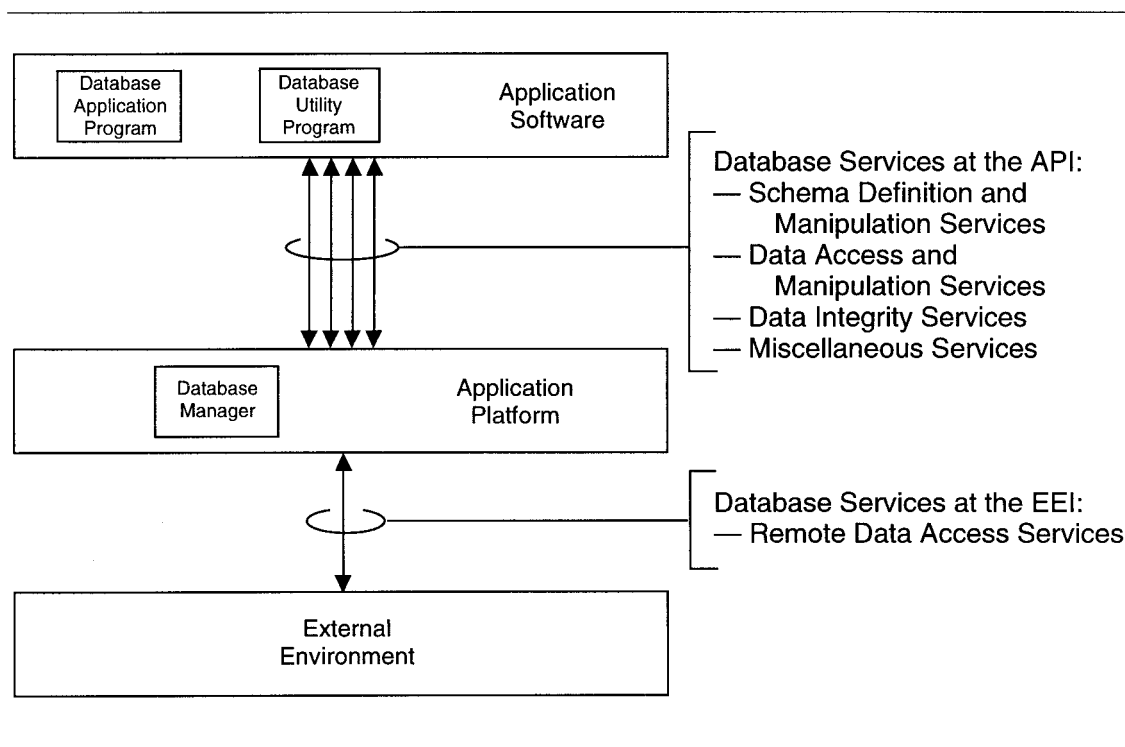


Figure 4.6—POSIX OSE Database Services Reference Model

The traditional database model can be incorporated into the POSIX OSE reference model, as in Figure 4.6. This depiction of the model shows the database manager as part of the application platform, which is available to the application through the OSE API.

The model depicted in Figure 4.6 describes the database services API as viewed by an application developer. The lines labeled “Database Services API” are discussed in 4.4.4.1. The following extensions to the model are specifically accommodated:

- There may be more than one database services API. For example, there may be an “SQL” API and an “ISAM” API.
- There may be more than one database manager implementation for each different API (for example, by two competing database vendors).
- Applications may access more than one database manager.

It should be noted that this model is very general and does not constrain the implementor. This model supports various implementation techniques, including

- An application and a database manager linked into a single process
- A database manager consisting of more than one process
- A database manager consisting of a client part linked into the application process and a server part running as a process on the same or another system

4.4.4 Services

The database manager described in the previous subclause provides services to the database application program via the database services API. The database utility programs provide other services (e.g., to human users such as a

database administrator). This subclause describes the database services needed by all users of the system. Database services are the specialized data services required to create, access, and manage databases located on a processor node. Users of these services include end users and those charged with the ongoing management of the information processing and database infrastructure.

4.4.4.1 API Services

This subclause describes the major categories of database services available at the POSIX API. These services include

- Schema definition and manipulation services
- Data access and manipulation services
- Data integrity services
- Miscellaneous services

The following paragraphs clarify that these services should be provided for a large number of objects, access methods, and types of database systems.

Types of data objects

The ability to perform the above operations on a variety of types of data objects, such as numeric data, text, graphics, image, documents, and voice.

Types of access methods

The ability to perform the above operations using a variety of access methods, such as indexed sequential access, nonindexed sequential access, and direct access.

Types of database management systems

The ability to perform the above operations on a variety of types of file and database management systems, such as hierarchical, relational, network, and object-oriented databases, and heterogeneous combinations of these database management systems.

4.4.4.1.1 Schema Definition and Manipulation Services

These services relate to the ability of the database administrator to define and alter database schemas. Schema definition is accessible to SQL programs through the more than 20 tables described in the information schema. The schema services are

Schema definition

Create table, create view, create assertion, grant privilege

Schema manipulation

Drop or add column, drop or add integrity constraint, revoke privileges

4.4.4.1.2 Data Access and Manipulation Services

These services relate to the ability of application programs to interrogate databases. These services are

Data access

Select row

Data manipulation

Insert, delete, or update data

Data query facilities

Specify search conditions, consisting of a combination of select lists, predicates, and comparison operators

Data transparency

Provide access to data, regardless of the location of that data

4.4.4.1.3 Data Integrity Services

These services relate to the ability of database management systems to protect the databases from hardware and software malfunctions. These services are

Locking

Specify locking of data to some degree of granularity

Consistency

Specify and execute check and referential constraints that help ensure data correctness

Transaction control

Specify commit and rollback commands and guarantee serializability for database transactions

Synchronous writes

Force the writing of data to nonvolatile storage

Deadlock detection and avoidance

Detect or avoid conditions where two or more processes are blocked because they require the same data simultaneously

4.4.4.1.4 Miscellaneous Database Services

Miscellaneous database services include

Privilege administration

Grant and revoke privileges for accessing and administering data.

Transaction management

Declare transactions at different levels of isolation. A transaction is a group of instructions that all have to succeed or fail at the same time; e.g., transferring money from a savings to a checking account.

Exception handling

Allow the operating system to notify the affected application so that the application may take remedial action when exceptions to normal processing occur.

Connection management

Make connections with different query language environments.

Reporting

Create formatted reports.

Dynamic facilities

Temporarily turn control of a database to the end user for interactive access and manipulation of data, and then return control to the application.

Data dictionary services

Get data about the data (i.e., metadata) stored in the database. This allows users and applications to use the database contents in a much more flexible way. These services allow a user to create, access, and manage this metadata much in the same way as other databases are maintained.

Dialogue Management

Initiate and terminate dialogues between systems.

4.4.4.2 EEI Services

EEI services are required for distributed database management systems. A common interchange format is also required to enable two or more databases to communicate with one another. The Remote Data Access (RDA) protocol standard specifies a protocol that allows remote access and updating of relational databases. RDA includes dialogue management, transaction management, resource handling, database language, and control services.

4.4.4.3 Database Resource Management Services

These services are not visible to the application programmer at the database services API. These services are usually provided by database utility programs. These services include

- Database administration services
- Database recovery services
- Distributed database management services
- Heterogeneous environment support services
- Checkpoint recovery services
- Flagger services

4.4.4.3.1 Database Administration Services

Database administration services are those used by a designated database administrator to structure and manage the configuration of a database as a whole. The administrator allocates resources and monitors utilization to assure that authorized users receive the proper services. Archive functions, journaling, and logging services may be provided to both database administrators and to general users.

4.4.4.3.2 Database Recovery Services

Database recovery services refer to the ability to notify the application or the database administrator that there has been a failure, to allow for recovery from failure, and to permit a slave copy to become a master copy.

4.4.4.3.3 Distributed Database Management Services

Distributed database management services support the partitioning and partial replication of the databases. The data administrator should have the ability to specify the physical location of data and structures in order to optimize performance and response times and to reduce costs.

4.4.4.3.4 Heterogeneous Environment Support Services

Heterogeneous environment support services permit local database systems to be of different types (e.g., inverted list, hierarchical, network, relational) by providing translators between the local database form and a general “network language.”

4.4.4.3.5 Checkpoint Recovery Services

Checkpoint recovery services support data integrity by allowing operations to be restarted from a previous known state.

4.4.4.3.6 Flagger Services

A flagger facility alerts programmers when application code does not conform to the standard in question. For example, the Federal Information Processing Standard (FIPS) version of the SQL standard incorporates a flagger facility to identify software features that are not compliant with the SQL standard and which may, therefore, result in code that is not interchangeable with other SQL-compliant database management systems.

4.4.5 Standards, Specifications, and Gaps

There are currently three related database standards, either completed or under development. These are the relational database language SQL, the Information Resource Dictionary System (IRDS) for data dictionary work, and an RDA protocol. See Table 4.10.

Table 4.11 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

4.4.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.4.5.1.1 Structured Query Language (SQL)

ISO/IEC 9075 :1992 {48} Database Language SQL

ANSI X3.168-1989 {109} Embedded SQL

ISO/IEC 9075 :1992 {48} provides many of the services described in 4.4.4, including Data Definition, Manipulation, and Integrity. It specifies two levels of compliance: the weaker Level 1 and the more capable Level 2. While ISO/IEC 9075 :1992 {48} deals with SQL independently of a programming language, ANSI X3.168-1989 {109} provides both procedural and embedded bindings of SQL to the programming languages COBOL, Fortran, Pascal, PL/I, C, and Ada.

Table 4.10—Database Standards

Service	Type	Specification	Subclause
Query Language:			
Schema definition and manipulation	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
Data access and manipulation	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
Data integrity	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
Transaction management	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
Exception handling	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
Connection management	S	ISO/IEC 9075 :1992 {48}	4.4.5.1
IRDS:			
IRDS requirements	S	ANSI X3.138-1988 {106}	4.4.5.1
IRDS command language and panel interface	S	ANSI X3.138-1988 {106}	4.4.5.1
IRDS export/import file format	S	ANSI X3.195-1991 {111}	4.4.5.1
IRDS framework	S	ISO/IEC 10027 :1990 {73}	4.4.5.1
IRDS services interface (includes data structures, service semantics, and Pascal binding)	S	ISO/IEC 10728 :1993 {89}, ANSI X3.185-1992 {110}	4.4.5.1
Services interface module for C language binding	S	ISO/IEC 10728 :1993 {89}	4.4.5.1
Services interface module for Ada language binding	E	Working Draft	4.4.5.2.1
IRDS design support for SQL applications	E	Working Draft	4.4.5.2.1
RDA:			
Dialogue management	S	ISO/IEC 9579 {55}	4.4.5.1
Transaction management	S	ISO/IEC 9579 {55}	4.4.5.1
Resource handling	S	ISO/IEC 9579 {55}	4.4.5.1
Database language	S	ISO/IEC 9579 {55}	4.4.5.1
Control	S	ISO/IEC 9579 {55}	4.4.5.1

NOTES:

1 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap.

2 — ISO has not yet assigned document numbers to the IRDS Service Interface Module bindings to the Ada programming language nor to the IRDS Design Support for SQL applications.

Work is currently planned by ANSI and ISO to include “generalized triggers,” “generalized assertions,” “recursive expressions,” “escape from SQL,” subtables, and support tools for object-oriented and knowledge-based systems.

Specifications are also emerging to extend existing ANSI and ISO SQL standards to support database interoperability. The SQL Access Group, a consortium of over 40 companies, is working to accelerate existing standards efforts through prototyping. To date, the group has produced two specifications:

- An API specification defining an embedded database language, known as SQL-89, based on the ANSI and ISO SQL definition
- A formats and protocol (FAP) specification for client/server communications, based on the ISO RDA SQL specification

These specifications augment the standards on which they are based. They define areas underlying the standards considered to be implementor defined.

Table 4.11—Database Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
SQL	G	S	G	G	S	G	S
IRDS	G	E	G	G	E	G	G

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
SQL	G	S	G	S	S	G
IRDS	G	G	G	S	G	G

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap
- 4 — APIs to the RDA standard are not defined.

4.4.5.1.2 Information Resource Dictionary System (IRDS)

ANSI X3.138-1988 {106}	Information Resource Dictionary System
ANSI X3.185-1992 {110}	IRDS Services Interface
ANSI X3.195-1991 {111}	IRDS Export/Import File Format
ISO/IEC 10027 :1990 {73}	IRDS Framework
ISO/IEC 10728 :1993 {89}	IRDS Services Interface

These standards are being developed by ISO/IEC JTC 1/SC21/WG3. Differences in scope and incompatibilities exist between the model being developed by ISO and the model approved by ANSI. ISO and ANSI are independently developing the Services Interface, and ANSI has developed an export/import facility, ANSI X3.195-1991 {111}.

In 1992, ASC X3H4 supported new work on the definition of a framework for the next-generation IRDS (IRDS2). This work is being harmonized with ISO in order to develop joint specifications for an IRDS that will meet future needs. This work continues to develop revisions that incorporate requirements from many different sources. An IRDS system service interface to the Pascal programming language is included in ISO/IEC 10728 :1993 {89}.

4.4.5.1.3 Remote Data Access (RDA) Protocol

ISO/IEC 9579-1 {55} Generic Remote Database Access

ISO/IEC 9579-2 {55} SQL Specialization

The RDA standard specifies a protocol that allows remote access and updating, via OSI communication protocols, of relational databases. ISO/IEC 9579-1 {55} provides a generic standard for remote database access and ISO/IEC 9579-2 provides the specific interface to SQL-compliant databases. The RDA standard could be used with database languages that are not SQL-compliant, but specialized standards would need to be developed to support these database languages.

RDA provides for the Data Transparency, Remote Data Access, and Support for Heterogeneous Environment requirements described in 4.4. This protocol is aimed at relational databases and other database types that provide access via relational interfaces such as SQL.

Much work is planned on in this area by ISO/IEC JTC 1/SC21/WG3. A specific area of current interest is a generic RDA that uses a nonspecific database language (i.e., not SQL).

4.4.5.2 Additional Specifications

4.4.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and may qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.4.5.2.1.1 SQL2 and SQL3

ISO/IEC DIS 9075-2, sometimes referred to as SQL2, provides for transaction management, including commit and rollback statements. Additionally, it includes some client/server operations, SQL sessions, and SQL connections, thus providing access to heterogeneous database management systems. SQL2 also improves diagnostic capabilities by including status parameters and supporting statements, and it adds support for dynamic SQL, which gives an application the ability to generate and execute SQL statements during application execution. ISO/IEC DIS 9075-3, sometimes referred to as SQL3, is projected for completion in 1995 and will provide a further extension to SQL.

4.4.5.2.1.2 SQL Ada Module Description Language (SAMeDL)

ISO/IEC 12227 :1995 {96}, SAMeDL, specifies a standard interface between the Ada programming language and SQL-compliant database management systems.

4.4.5.2.1.3 IRDS

In addition to the IRDS system service interface to the Pascal programming language defined in ISO/IEC 10728 :1993 {89}, IRDS system service interfaces to the Ada and C programming languages are being developed. An IRDS standard is also being developed by ISO to provide design support for SQL applications.

4.4.5.2.2 Public Specifications

X/Open, together with the SQL Access Group, has published a profile of the ISO/IEC 9075 :1992 {48} SQL standard, which documents a stepping-stone towards full implementation of the standard and reflects the plans of database systems providers.

X/Open: Structured Query Language (SQL) {B60}

X/Open: SQL Remote Database Access {B68}

4.4.5.3 Unaddressed Services

There are several significant areas described in 4.4.4 as services that are not addressed by standards:

- Methods to access data, such as hashing, and indexed sequential access to data are not currently standardized.
- Standardization of object-oriented models has also not taken place, although X3 has addressed the technical area (see B.2.3.7). A Database Systems Study Group (DBSSG) completed a study in 1991 summarized in a report entitled, *Object Oriented Database Task Group Final Report* (or OODBTG Final Report). Based on the report, a new technical committee was formed under X3 to examine object technology for further standardization. The X3H7 Object Information Management (OIM) is currently investigating the feasibility of standardization in this area.
- API and EEI standards are needed for the management and control of distributed databases. For example, standards are needed to ensure the integrity between the data residing at two or more locations.

4.4.6 POSIX OSE Cross-Category Services

4.4.6.1 Security

The ability to specify logical database access control mechanisms is important to database security. Database integrity, confidentiality, and availability are also important, as are accountability for access and modifications, and support for multilevel security.

4.4.7 Related Standards

There are several areas closely related to the database area that are worth considering with respect to standardization.

The first area to consider is the communications and networking area. Interoperability for distributed applications or the use of distributed databases may indicate the use of communication software adhering to networking standards. See 4.3 for further discussion of that area. The following standards described in that clause are specifically related:

ISO/IEC 9804 :1994 {65}	OSI CCR services
ISO/IEC 9805-1 :1994 {66}	OSI CCR protocol
ISO/IEC 8824 :1990 {41}	ASN.1 specification
ISO/IEC 8825 :1990 {42}	ASN.1 basic encoding rules

The second area to consider is transaction processing. That area goes further in addressing the total requirements for distributed applications. See 4.6.

4.5 Data Interchange Services

4.5.1 Overview and Rationale

The data interchange services components of the POSIX OSE provide specialized support for the exchange of data between applications or components of applications. Without support for data interchange, problems can arise when attempts are made to move data between different operational environments or between two related applications. The exchange of data may occur through many mechanisms, including plain files, network messages, or database fields. More specifically, data interchange problems arise in each of the four following situations:

- Movement of a single application program and its associated data between operational environments
- Movement of data between cooperating application software within the same operational environment
- Movement of data between cooperating application software operating in differing operational environments
- Movement of data between related, but not cooperating, application software within a single operational environment, and across differing operational environments

From the global view, standards for data interchange services and formats can provide the means to satisfy the needs of each of these situations. These standards need to define character sets and data representation, data formats, and data descriptions that are consistent across all implementations of the POSIX OSE.

4.5.2 Scope

The data interchange services and formats of the POSIX OSE include standard services, protocols, and data formats required to ensure that structured data, including regular text, structured text, vector graphics, and raster graphics, can be interchanged between related application software. Physical media formats are beyond the scope of the POSIX OSE.

4.5.3 Reference Model

The data interchange services relate directly to the POSIX OSE reference model in Figure 3.1. Figure 4.7 shows the components of the reference model that are significant for data interchange. The reference model defines the conceptual relationships required to provide these facilities. It should not be viewed as a description of an implementation. To provide the data interchange services, the POSIX OSE has to permit application software to transfer data to and from the external environment.

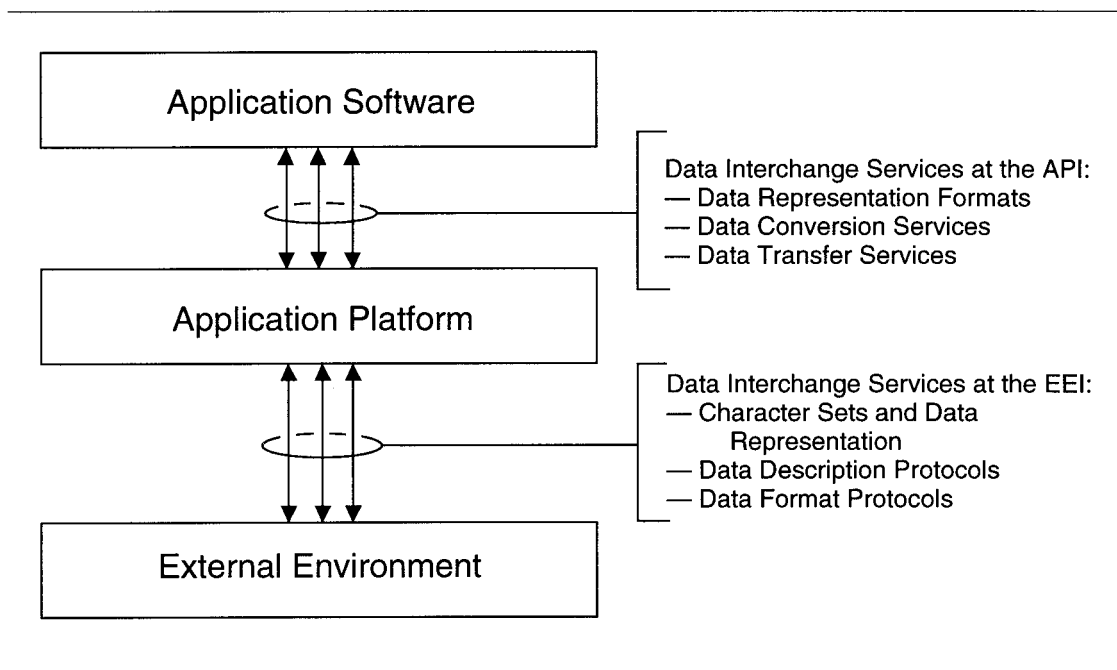


Figure 4.7—POSIX OSE Data Interchange Services Reference Model

The application software requests this transfer through the API. In response to those requests, the data interchange components of the application platform handle conversions to and from standard formats and the transfer of the information across the EEI. The EEI, which defines the format specifications required to support data interchange, can be divided into data description protocols and data format protocols. Data description protocols provide a means to identify the data that is present. Data format protocols provide the storage representation of the actual data.

Currently, this model is only partially supported by standards. Fundamental data representation is fairly well standardized. Some work is beginning on data format protocols standards, particularly in the networking area. At this time, no general standards exist to support data description protocols.

4.5.4 Services

This subclause details the data interchange services and protocols that support application portability and interoperability. API services are described in 4.5.4.1. EEI services are described in 4.5.4.2.

Data interchange is one of the components of the POSIX OSE that is now just beginning to evolve. Most existing practice is limited to specific application domains or development environments. As a developing area, data interchange represents gaps in both the definition of services and standards. The data interchange component is, nonetheless, critical for supporting application portability and interoperability. The data interchange services are described to the extent possible at this time in their evolution. More detail will be added in future revisions of this guide.

4.5.4.1 API Services

The API services to support data interchange need to provide the ability to represent, convert, and transfer data across application program domains using the formats and protocols provided at the data interchange EEI.

Currently, little work has been directed at defining API-level services for data interchange. Data interchange API services need to provide a means to request that specific data be represented using the EEI services defined below. Progress in this area is following a pattern similar to the development of networking standards. That is, initial standards defined protocols and only after those were in use has attention shifted to providing a standard mechanism for requesting those networking services.

4.5.4.2 EEI Services

This subclause identifies the EEI services that support data interchange. These services are all in the form of protocol and format definitions. As shown in Figure 4.7, these services include

- Character sets and data representation
- Data format protocols
- Data description protocols

These services support the exchange of data between application software entities, both within a single application platform and between application platforms.

4.5.4.2.1 Character Sets and Data Representation

The ability to support character sets and data representation is crucial to providing effective data interchange between application software operating under differing language and cultural conventions. These services add facilities to the POSIX OSE to identify the character set and data representations associated with any particular data. A detailed description of the services in this area can be found in 5.1.4.1.1.

4.5.4.2.2 Data Format Protocols

The data format protocols need to provide the ability to identify the representation of the data in a manner that is independent of the specific execution environment. The data format protocol layer adds attributes that describe the fundamental characteristics of the data that have to be known to retrieve the data value properly, given the storage formats that are native on the hardware/software environment where the data is used. The complete attribute information required to decipher that data value includes

- Detailed storage format for the value
- The data value in an environment-neutral format

The data format protocols protect applications from hardware/software differences between environments. Specifically, the protocols ensure that data remains stable when moving between environments where the character set, word size, or byte ordering may differ.

4.5.4.2.3 Data Description Protocols

Data description protocols provide the ability to share data between related application software entities, even if they were not specifically written to cooperate. Building upon the facilities provided by the previous two data interchange EEI services, data description protocols provide a means of associating a name or other identifier with the individual data elements in a standard manner. This permits an application program to identify correctly data that was created by an unrelated application. To date, most standards in this area have limited themselves to specific application areas, and no general solution has been provided.

4.5.5 Standards, Specifications, and Gaps

See Table 4.12.

4.5.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.5.5.1.1 Office Document Architecture (ODA)/Office Document Interchange Format (ODIF)/Office Document Language (ODL)

The ODA standard (ISO 8613 {28}) enables users to interchange the logical structure, content, presentation style, and layout structure (the physical appearance) of documents from one application to another, or from an application to various output devices. ODIF is an ASN.1 (ISO/IEC 8824 : 1990 {41} and ISO/IEC 8825 : 1990 {42}) encoding for documents suitable for interchange between applications. ODL is a Standard Generalized Markup Language (SGML) encoding of ODA documents. ODA documents represented by SGML are interchanged using SGML Document Interchange Format (SDIF) (ISO 9069 : 1988 {47}).

4.5.5.1.2 Standard Generalized Markup Language (SGML)/SGML Data Interchange Format (SDIF)

SGML (ISO 8879 : 1986 {44}) is a markup language for defining the logical structure of documents. Markup consists of special characters to denote the structure of a type of document, the meaning being defined in the document type definition. SGML provides a means to specify what is allowed on the particular markup, what is required, and how the markup is distinguished from text. While the standard specifies that there shall be markup, its meaning is left to the application. SDIF (ISO 9069 : 1988 {47}) enables the interchange of ODA documents.

Table 4.12—Data Interchange Standards

Service	Type	Specification	Subclause
Data Format Protocols:			
Document representation	S	ISO 8613 {28} (ODA/ODIF/ODL), ISO 8879 : 1986 {44}, ISO 9069 : 1988 {47} (SGML/SDIF)	4.5.5.1
Hyperdocument representation	S	ISO/IEC 10744 : 1992 {90} (HyTime)	4.5.5.1
Graphics representation	S	ISO/IEC 8632 : 1992 {29} (CGM), ANSI/ASME Y14.26M-1989 {112} (IGES)	4.5.5.1
Electronic data representation	S	ISO 9735 : 1988 {64} (EDIFACT)	4.5.5.1
Font information representation	S	ISO/IEC 9541 {52} (Fonts)	4.5.5.1
Product data representation	S	ISO/IEC 10303 {83} (STEP)	4.5.5.1
Circuit data representation	S	IEEE Std 1076-1993 {122} (VHDL)	4.5.5.1
Document format	S	ISO/IEC DIS 10179 {79} (DSSSL), ISO/IEC DIS 10180 {80} (SPDL)	4.5.5.1
Spatial data representation	E	FIPS Publication 173 {B20} (SDTS)	4.5.5.2.1
CASE data representation	E	EIA/IS-106 {B10}, EIA/IS-107 {B11}, EIA/IS-108 {B12}, EIA/IS-109 {B13}, EIA/IS-110 {B14}, EIA/IS-111 {B15}	4.5.5.2.1
Data Description Protocols:			
Document interchange	S	ISO 8613 {28} (ODA/ODIF/ODL), ISO 8879 : 1986 {44}, ISO 9069 : 1988 {47} (SGML/SDIF), ISO/IEC DIS 10180 {80} (SPDL)	4.5.5.1
Graphics interchange	S	ISO/IEC 8632 : 1992 {29} (CGM), ANSI/ASME Y14.26M-1989 {112} (IGES)	4.5.5.1
Electronic data interchange	S	ISO 9735 : 1988 {64} (EDIFACT)	4.5.5.1
Font information exchange	S	ISO/IEC 9541 {52} (Fonts)	4.5.5.1
Product data exchange	S	ISO/IEC 10303 {83} (STEP)	4.5.5.1
Spatial data interchange	E	FIPS Publication 173 {B20} SDTS)	4.5.5.2.1
CASE data representation	E	EIA/IS-106 {B10}, EIA/IS-107 {B11}, EIA/IS-108 {B12}, EIA/IS-109 {B13}, EIA/IS-110 {B14}, EIA/IS-111 {B15}	4.5.5.2.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.5.5.1.3 Hypermedia

Hypermedia/Time-based Structuring Language, or HyTime, is the standard (ISO/IEC 10744 :1992 {90}) that defines a model and language for the representation of “hyperdocuments” that link and synchronize static and time-based information contained in multiple conventional and multimedia documents and information objects. HyTime is parsable as SGML.

4.5.5.1.4 Computer Graphics Metafile (CGM)

CGM (ISO/IEC 8632 :1992 {29}, also ANSI X3.122-1986) provides a standard means for the storage and exchange of graphics data. Graphics data is stored in a device- and resolution-independent fashion that can support the display of the data. Pictures are described in CGM as a collection of elements of different kinds, representing, for example,

primitives, attributes, and control information. Part 1 of the standard contains the semantics of all the elements. Parts 2–4 contain the syntax of three different bindings of the standard: character-coded, binary, and clear-text encodings.

4.5.5.1.5 Electronic Data Interchange (EDI)

The EDI standard [ISO 9735 :1988 {64} (EDIFACT)] provides support for the exchange of structured business data intended for automatic processing. EDI is typically used to transfer business documents such as purchase orders, invoices, and electronic funds transfer information, but can be used for many other applications where the data to be transferred can be formatted into a message of fixed structure.

4.5.5.1.6 Initial Graphics Exchange Specification (IGES)

IGES (ANSI/ASME Y14.26M-1989 {112}) is used heavily in the exchange of technical drawings, documentation, and other data required for product design and manufacturing, including geometric and nongeometric data such as form features, tolerances, material properties, and surfaces. The information typically associated with computer-aided design (CAD), and computer-aided manufacturing (CAM) can be described.

4.5.5.1.7 Font Information Interchange Standard

ISO/IEC 9541-1 :1991 {52} defines a method of naming glyphs and glyph collections in a manner that is independent of any coding technique. ISO/IEC 9541-2 :1991 {52} provides the interchange formats as well as the minimum subsets of the information required for interchange.

4.5.5.1.8 Standard for the Exchange of Product Model Data (STEP)

STEP (ISO/IEC 10303 {83}) is a nonproprietary mechanism capable of completely representing product data throughout the life cycle of a product. The completeness of this representation makes it suitable not only for file exchange, but also as a basis for implementing and sharing databases of archived information. STEP defines standards for data content, a description language (EXPRESS), a file interchange format, and an API (SDAI).

4.5.5.1.9 Hardware Description Language (VHDL VHSIC)

The VHDL standard (IEEE Std 1076-1993 {122}) defines a representation for the exchange of CAD representations of electronic circuits.

4.5.5.2 Additional Specifications

4.5.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.5.5.2.1.1 Document Style Semantics and Specification Language (DSSSL)

DSSSL (ISO/IEC DIS 10179 {79}) defines the semantics and syntax of a language for the specification of document processing. The semantics include a document architecture for typographic presentation style and other document processing specifications typically associated with traditional text processing languages. It is designed to specify the processing of documents conforming to SGML.

4.5.5.2.1.2 Standard Page Description Language (SPDL)

SPDL (ISO/IEC DIS 10180 {80}) defines a language for representing images that are to be displayed on a screen, printed on an output device, or transmitted through communications media from one application to another. To support the interchange of SPDL documents in a variety of environments, SPDL provides two document representation formats: a binary interchange format and a clear text interchange format.

4.5.5.2.1.3 Spatial Data Transfer Specification (SDTS)

SDTS provides specifications for the organization and structure of digital spatial data transfer, definition of spatial features and attributes, and data transfer encoding. It is a United States Federal Information Processing Standard (FIPS Publication 173 {B20}) developed by the United States National Committee for Digital Cartographic Standards, headed by the US Geological Survey (USGS), Department of the Interior. USGS is preparing to submit the SDTS to ANSI for promotion as an ANSI standard and then to ISO for promotion as an international standard.

4.5.5.2.1.4 CASE Data Interchange Format (CDIF)

The CDIF Technical Committee is developing a data interchange format to serve as an industry standard for exchanging information between computer-aided software engineering (CASE) tools. CDIF is an EIA-endorsed initiative resulting in a set of CDIF interim standards (EIA/IS-106 {B10} , EIA/IS-107 {B11} , EIA/IS-108 {B12} , EIA/IS-109 {B13} , EIA/IS-110 {B14} , and EIA/IS-111 {B15}). It assumes that two or more tools may interface asynchronously with each other and will transfer information from one to another via "CDIF files." The types of information that may be contained in these files is defined by the CDIF Conceptual Models.

4.5.5.2.2 Public Specifications

None.

4.5.5.3 Unaddressed Services

None of these standards addresses a general means to handle application data in a manner to ensure portability between environments. The existing standards have all evolved to support the interchange of specific types of data between separate applications.

4.5.6 POSIX OSE Cross-Category Services

4.5.6.1 Internationalization Services

The POSIX OSE cross-category services relevant to data interchange services include those that address character sets and data representation. Character sets and data representation services include the capability to store, manipulate, and retrieve data independent of the coding scheme used. This includes the capability to access a central character set repository of all coded character sets used throughout the platform. Character sets will be uniquely identified so that an interchange application can select the coded character set to be used. Also included is the capability to recognize the coded character set of data entities.

4.5.7 Related Standards

The following standards are related to the services covered in this clause as they address some of the services described in 4.5.4 at some level. Each of these related standards is addressed fully in 4.3.

ISO/IEC 8824: 1990	ASN.1 specification
ISO/IEC 8825 : 1990 {42}	ASN.1 basic encoding rules
ISO/IEC 10021 {71}	X.400 Message Handling System (MHS)

4.6 Transaction Processing Services

4.6.1 Overview and Rationale

The database management clause (see 4.4) describes some transaction processing (TP) services (specific to databases). This clause describes the complete set of transaction processing services from the application software point of view. Note that transaction processing services have long been regarded, variously, to be within the domain of databases or within the domain of operating systems and more recently within the domain of communication software. These services are more broadly applicable than just these three areas, and so are treated here as a separate clause.

Transactions (“units of work”) have boundaries (start points and end points) that are determined by the action of the transaction application program. The transaction application program can request either to commit or roll back the work done in the transaction when it identifies the end point. The system will complete a commit operation only if all operations performed during the transaction can complete successfully. Otherwise, the system will abort the transaction (roll back the work done by it) and notify the transaction application program of this action.

The following is quoted with a few editorial changes from ISO/IEC 10026-1 :1992 {72}, the ISO Distributed Transaction Processing (DTP) standard:

A transaction is characterized by the ACID properties: atomicity, consistency, isolation, and durability
 Atomicity implies that the operations of a unit of work are either all performed, or none of them are performed
 Consistency implies that the operations of a unit of work, if performed at all, are performed accurately, correctly, and with validity, with respect to application semantics
 Isolation implies that the partial results of a unit of work are not accessible, except by operations which are part of the unit of work, and also implies that units of work which share bound data are serializable
 Durability implies that all the effects of a completed unit of work are not altered by any sort of failure

4.6.2 Scope

This clause deals with the transaction processing services needed for a large number of types of transaction processing including the following:

- Transactional access to a single database manager on a single machine
- Transaction access to nondatabase *resource managers* (such as the software managing the cash in an automatic teller machine)
- Distributed Databases — databases spanning multiple machines, but accessed by application programs as if just a single database
- Online Transaction Processing (OLTP) — the scheduling of *transaction programs* based on terminal input with consolidated recovery of the database updates and the terminal messages
- Distributed Transaction Processing — different machines running multiple application programs with multiple databases, using a client/server or conversational application-to-application communication paradigm

Note that transaction processing services are used in all of the above situations, and others, too.

Finally, it should be noted that *transactions* are not really *messages*, but rather “units of work” that may encompass multiple messages. Furthermore, while *transaction processing* has traditionally been synonymous with “OLTP,” where so-called “immediate transactions” are the norm, other types of transactions are also covered: *batch transactions* (where the work is done in the background) and *queued transactions* where there may be a time dependence on the transaction, such as a fixed start time.

4.6.3 Reference Model

This subclause addresses the transaction processing reference model, the POSIX OSE reference model (incorporating transaction processing considerations), and other important considerations introduced by transaction processing.

4.6.3.1 Transaction Processing Reference Model

A model for transaction processing is developed in this guide to complement the POSIX OSE reference model. Current work in progress within groups such as ISO/IEC JTC 1/SC21 (OSI—DTP) may result in a transaction processing reference model more suitable than the one developed in this guide. At that time, such a model will be referenced and incorporated into the POSIX OSE reference model. Until that time, the current model will be used as a convenient means for describing the services needed in this domain.

While transaction processing services have usually been thought of as applying to databases, the applicability goes further. Nonetheless, the description given here of the transaction processing model shows how the transaction processing program can view the transaction services as an extension of the database view of the POSIX OSE reference model, as shown in Figure 4.5. From the transaction application program point of view, a transaction processing system has additional capabilities that go beyond those provided by database systems. These services to the transaction application program are provided at an API that is called the *transaction manager API*. (See Figure 4.8.) For convenience in discussing the model, the provider of those services is called the *transaction manager* (TM).

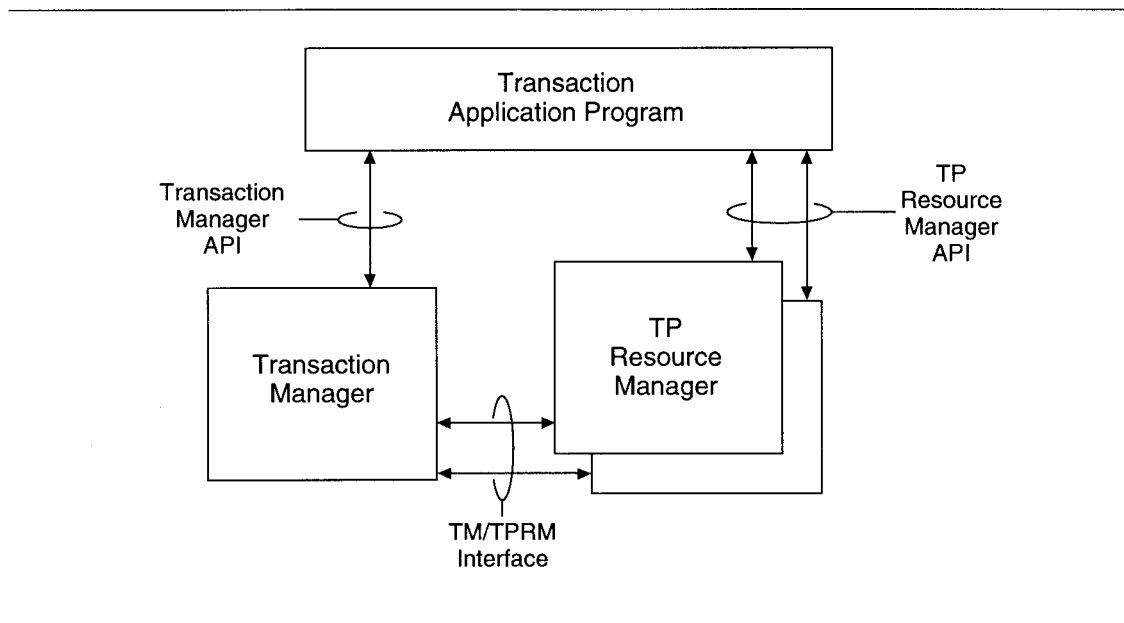


Figure 4.8—Transaction Processing Model

The transaction application program requests services provided by the *TP resource manager*¹⁷ (e.g., a database manager) via the *TP resource manager API*. The transaction manager API and the TP resource manager API are called the *transaction services API* and provide all the services needed by transaction application programs.

¹⁷The term *TP resource manager* should not be confused with a different term, *resource management services*, which is a type of service described in most service category classes in this section (e.g., 4.6.4.3).

The ACID properties (see 4.6.1) are maintained for each managed resource by a *TP resource manager (TPRM)*, coordinated by a *transaction manager*. The interface between the TP resource manager and the transaction manager is called the *transaction manager/TP resource manager interface (TM/TPRM interface)*.

The ACID properties can be applied not only to resources such as databases, but also to other resources that might not be obvious. For instance, a transaction that dispenses cash may wait until the cash dispensing machine has signaled completion before considering the transaction complete and updating involved accounts. This illustration also shows the limits of transaction processing resource management. The machine could signal completion, but a mechanical problem could prevent the cash from being dispensed correctly, undetected by the system.

Besides database TPRMs and miscellaneous nondatabase TPRMs, a third class of TPRMs exist: Communication TPRMs (cTPRM). Services provided by cTPRMs are used when two cooperating transaction application programs need to communicate with each other in the context of the same transaction. Figure 4.8 is the simplest drawing of the model. To show communication in distributed transactions, cTPRMs would also be shown. Multiple instances of the model would be shown (one for each cooperating transaction application program), and lines would connect the cTPRMs, showing that the cTPRMs provide a communication path for the transaction application programs. At least two communication paradigms have been identified as beneficial to cooperating transaction applications programs: client/server (RPC, single request/response) and conversational (peer-to-peer, dialogue).

4.6.3.2 POSIX OSE Reference Model (with Transaction Processing)

The transaction processing model is shown integrated into the POSIX OSE Reference Model in Figure 4.9.

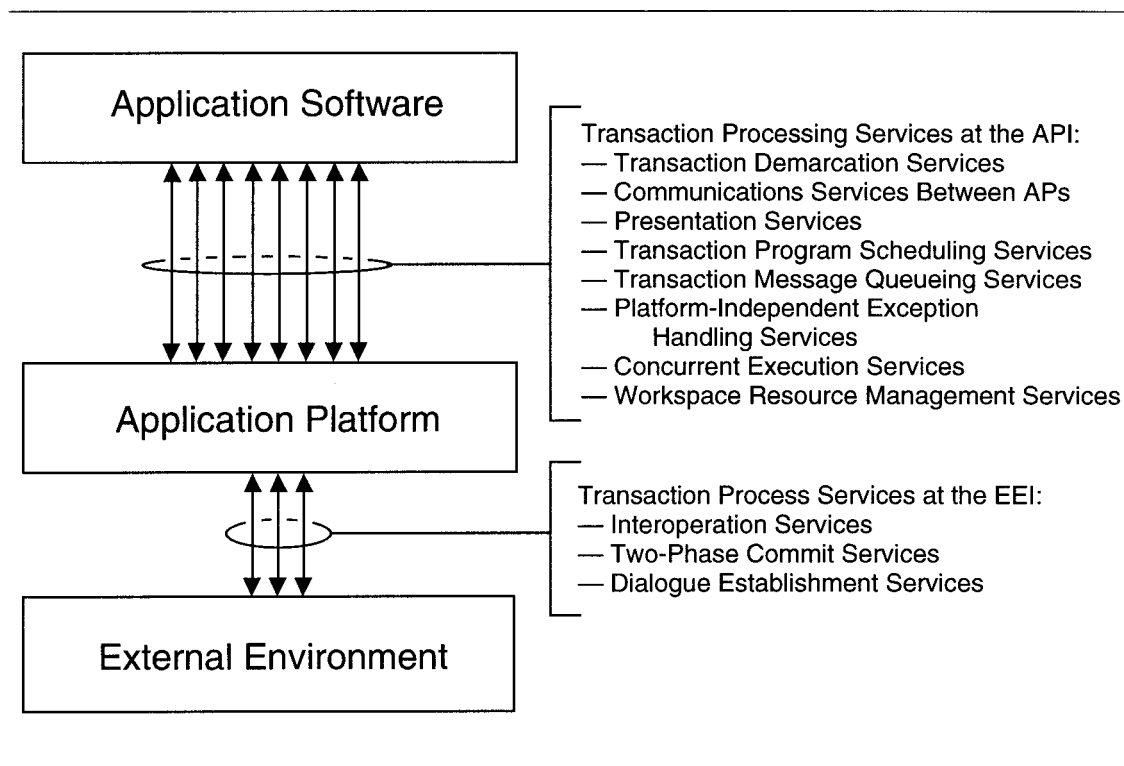


Figure 4.9—POSIX OSE Transaction Processing Services Reference Model

4.6.4 Services

Services provided via the transaction processing services API are described in 4.6.4.1. Services to enable the distribution of transaction processing are described in 4.6.4.2. General services, mostly performing administrative functions, are described in 4.6.4.3.

4.6.4.1 API Services

The transaction processing services API provides various services to the application programmer:

Transaction demarcation

- Indicate the start of a transaction.
- Indicate a transaction has ended successfully (commit) or unsuccessfully (rollback).
- Suspend and resume transaction mode (do work that is not committed or rolled back when the transaction is completed). This can be thought of as nesting nontransaction work within a transaction.

Communications with transaction processing application programs

- Call another transaction application program (possibly remote) within the context of a transaction.
- Open a dialogue, initiate a transaction, and send and receive messages to and from another transaction application program (possibly remote) within the context of a transaction.

NOTE — The above services provide *Distributed Transaction Processing*.

Presentation services

- Communicate with terminals via a record-oriented, device-independent presentation service interface. This enables transaction application programs to be device independent.
- Send and receive messages to and from terminals within the context of a transaction (e.g., messages sent to terminals are not to be actually delivered unless the transaction commits).

Transaction program scheduling—immediate

- Cause the start of a transaction application program from an application program that is not part of the transaction. Involved here are two transactions: one starts the other. The actual scheduling of the second transaction may, or may not, be dependent on the completion of the original transaction.

Transaction program scheduling—batch and deferred

- Cause a request for execution of another transaction program to be put on a queue; i.e., batched. Involved here are two transactions; one starts the other. The actual scheduling of the second transaction can be dependent on the completion of the original transaction.
- Allow for specification of start times with queued requests; i.e., deferred requests.
- Allow for prioritization of queued requests.

NOTE — The actual handling of messages may be dependent on the completion of the original transaction.

Platform-Independent exception handling

- Inform transaction application programs of system and application exceptions via a set of platform-independent exception classes, types, and codes. This enables truly portable transaction application programs.

Concurrent execution

- Cause two or more segments of a transaction application program to execute in parallel.

Workspace resource management

- Support in-memory data structures and variables that are committed or rolled back with the transaction. This enables easy rollback and restart of a transaction application program.

NOTE — Several of these services are similar to, but semantically different from, similar sounding services in other clauses of this section. They are listed here because they are "transactional"; i.e., the concept of a transaction and the ACID properties are provided by these services.

TP resource managers provide services usable by the transaction application program and are made visible by the TP resource manager API. An example of this is the Database API services; see 4.4.4.1.

NOTE — TP resource managers, in general, "protect" a critical resource. Databases are good examples of TP resource managers where the resource actually being protected is the data. Often the data reflects the amount of a real resource, such as cash holdings. In this case, a tangible resource is indirectly protected by a TP resource manager. The importance to the

enterprise in insuring that the data (quantifying money) is accurate should be obvious. Other TP resource managers, on the other hand, could protect an actual, tangible resource. An example of such a TP resource manager is the program that controls the cash drawer of an automated teller machine. The resource protected is the cash in the drawer. The actual API of the TP resource manager protecting that resource could include the ability to reduce the amount of money in the drawer (by pushing it out of the machine). A transaction application program using two TP resource managers (a conventional database manager that keeps track of the balance in accounts, and the cash drawer TP resource manager of the teller machine) would want to insure that the two TP resource managers decrement both the cash and the balance of the correct account in the context of a single transaction (i.e., with the ACID properties.)

The TP resource manager API, then, generally provides the following services:

- Increments or decrements a valuable resource by a certain amount.
- Determines the amount of a valuable resource that remains.

Specific capabilities for the very wide variety of specific TP resource managers cannot be documented in this guide.

4.6.4.2 EEI Services

When two or more machines in a distributed environment are involved in the same transaction, the following service is required:

- The ability for two application platforms to interoperate with each other (pass along global transaction identifiers, participate with each other in commitment process, participate with each other in recovery).

4.6.4.3 OLTP Resource Management Services

The services listed in this subclause are not provided by APIs or EEIs.

Management services

Control the operation of the transaction processing services, including the ability to assign dispatching priorities to individual transaction application programs.

Monitoring services

Collect data on resource utilization for purposes such as performance analysis and accounting (data on utilization of the transaction processing services resources: processes, connection pools, etc.).

Modeling services

Predict the required system resources and expected performance to process a given transaction processing workload.

Directory/Namespace services

Map names to addresses.

Recovery/Restart services

Recover and restart transactions involving one or more transaction application programs using one or more TP resource managers.

Test services

Automatically generate tests for workload simulation, etc.

System configuration services

Replace or add transaction application programs without the need to shut down the execution environment; add services to manage association (connection) pools.

Conformance classes

Define formal subsets of the OLTP functionality so both a PC/workstation implementation and a host/server implementation can claim conformance. The functionality needed on an OLTP workstation is different from that needed on an OLTP host.

4.6.5 Standards, Specifications, and Gaps

See Table 4.13.

Table 4.13—Transaction Processing Standards

Service	Type	Specification	Subclause
Transaction demarcation	G	n/a	4.6.5.3
Communications between transaction application programs	G	n/a	4.6.5.3
Presentation	G	n/a	4.6.5.3
Transaction program scheduling	G	n/a	4.6.5.3
Transaction message queuing	G	n/a	4.6.5.3
Platform-Independent exception handling	S	ISO/IEC 10026-1 :1992 {72}	4.6.5.1
Concurrent execution	S	ISO/IEC 10026-1 :1992 {72}	4.6.5.1
Management	G	n/a	4.6.5.3
Monitoring	G	n/a	4.6.5.3
Modeling	G	n/a	4.6.5.3
Directory/Namespaces	G	n/a	4.6.5.3
Recovery/Restart	G	n/a	4.6.5.3
Test	G	n/a	4.6.5.3
System configuration	G	n/a	4.6.5.3

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.14 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

4.6.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

Table 4.14—Transaction Processing Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
n/a							

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
n/a						

NOTES:

1 — LIS—Language-independent specification is available.

2 — Ada, APL, Basic, ...—Language-dependent specification is available.

3 — S, E, P, G—Standard, Emerging Standard, Public Specification, Gap

4 — When standard APIs are defined for transaction processing, this table will be populated in later versions of this guide.

4.6.5.1.1 OSI DTP

ISO/IEC10026 {72}

These standards (Parts 1-3), which are being developed by ISO/IEC JTC 1/SC21/WG5, deal expressly with the OSI services and protocols for transaction mode communications in an OSI environment. They provide for some of the communication services described in 4.6.4.1.

4.6.5.2 Additional Specifications

None.

4.6.5.3 Unaddressed Services

4.6.5.3.1 X/Open TP

The X/Open reference model for transaction processing contains a functional description of a software architecture that allows the work of multiple resource managers to be coordinated by a transaction manager. The XA specification describes a simple interface between a transaction manager and a resource manager.

X/Open: Distributed TP: Reference Model {B66}

X/Open: Distributed TP: The XA Specification {B59}

Other than the work of X/Open TP, the following areas are not currently being addressed by formal standardization or public specification activities: communications, terminal communications, program scheduling, monitoring, modeling, directory/namespace, and system configuration.

4.6.5.3.2 EWOS, OIW, AOW

These regional workshops all have groups working on profiles centered on ISO/IEC 10026 {72}. See Annex B for further information on these organizations.

4.6.6 POSIX OSE Cross-Category Services

Not applicable.

4.6.7 Related Standards

4.6.7.1 Commitment, Control, and Recovery (CCR)

The following standards concerning commitment are related to the ISO/IEC 10026 {72} series and are referenced in it:

ISO/IEC 9804 :1994 {65}	OSI CCR services
ISO/IEC 9805-1 :1994 {66}	OSI CCR protocol

4.6.7.2 Structured Query Language (SQL)

The following standards for SQL also provide transaction demarcation services for relational database access:

ISO/IEC 9075 : 1992 {48}
ANSI X3.168-1989 {109}

See 4.4.

4.7 User Command Interface Services

4.7.1 Rationale and Overview

Although system-level services are necessary for application portability and interoperability, they are insufficient for the system needs of many users. To maximize portability, many users also require the commands, command interpreter (shell), compilers, editors, and other utilities that have been traditionally associated with many operating environments. These user command interface services facilitate the successful porting of applications and help users to manage and maintain applications and to solve problems on an ad hoc basis. The standardization of these utilities allows users and programmers to move from platform to platform without having to relearn the user command interface for each application platform.

4.7.2 Scope

This clause describes how a user interacts with an application platform by executing general-purpose user commands. This user command interface is also available to applications so that applications can execute user commands. A standardized user command interface provides a consistent, interactive environment across platforms for users and programmers.

Commands that are outside the scope of this clause are

- System administration and installation commands; see 5.3
- Text formatting programs; see 4.5
- Database commands; see 4.4
- Networking and communication commands; see 4.3
- Graphical user interfaces; see 4.9
- Software development support commands; see 4.11

4.7.3 Reference Model

The use of the user command interface services presented in this clause is consistent with the reference model in Section 3. The OSE reference model for the user command interface also is consistent with typical implementations for user command languages in traditional UNIX-based systems.

As Figure 4.10 shows, the user command interface is available both to users (through the EEI) and to applications (through the API). Any operating system implementation can reside underneath the APIs and EEIs.

The API and EEI user command interfaces provide access to a software component (known as a command interpreter or shell) that interprets the commands issued by either the user or the application. The command interpreter acts as an intermediary between the command API and EEI and the system-level services of the base application platform. The command interpreter reads the commands entered and parses them. Depending on the type of command (e.g., utility or built-in shell command), the command interpreter either executes the command for the user or application, using the system-level services of the base application, or it calls on the system-level services to create a new process that executes the command.

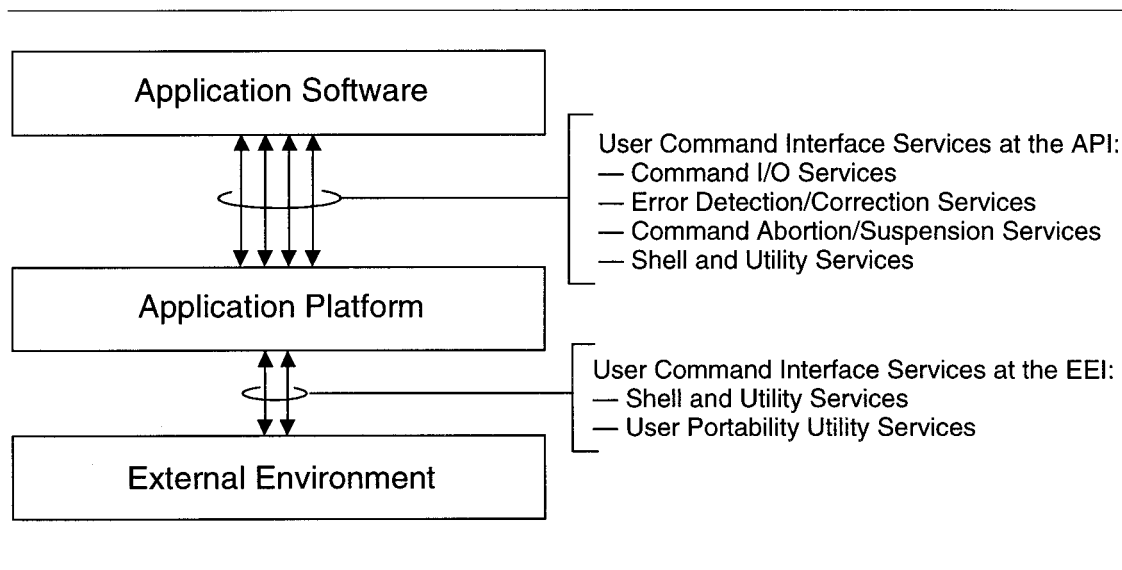


Figure 4.10—POSIX OSE User Command Interface Services Reference Model

None of the methods of executing commands have an impact on the API or EEI specifications.

The user command interfaces may be available to users and applications either locally or remotely. Remote invocation of the command interfaces of a system is provided through networking and data interchange capabilities. These are described in 4.3 and 4.5. Alternatively, remote access to the user command interfaces of a system may be available through certain interapplication services.

4.7.4 Services

There are three major aspects of user command interface services that need to be addressed for practical support of multivendor application portability and system interoperability. The first aspect consists of the basic functionality and interfaces provided for generally usefulness. The second aspect concerns the ability to move applications, such as script files, between platforms. The third aspect concerns user portability so that the same user command interface is available on different platforms.

Since most user command interfaces are available at the API and EEI, the API and EEI services are very similar. This clause, therefore, discusses primarily the EEI user command interface services. The API service subclause discusses only the additional services for applications. The description of API and EEI services in 4.7.4.2 discusses scripting language services, which are applicable to both the API and EEI.

4.7.4.1 API Services

In a user command API, the output syntax of the commands and command responses (such as error messages) need to be standardized, in addition to the calling sequence and allowable inputs. Such standardization is necessary to allow applications executing a command to parse reliably the output of that command.

The API should be able to access all of the services available to the user at the EEI. The additional services for the API give the application the ability to

- Provide the input to the command and access the output of the command when necessary
- Obtain additional data or commands from a file
- Detect and correct errors as the command is executed
- Abort or suspend the command as it is executing

4.7.4.2 API and EEI Services

It is also important to have the ability to create script files that are combinations of commands. The scripting language developed for this purpose is an application development language known as the shell. The scripting language includes the following services:

- Execute commands conditionally
- Execute commands repeatedly
- Display output
- Prompt the user for input
- Execute commands and obtain error information

The services and standards for the scripting language are described in this clause, rather than 4.1 (languages), because they are so closely related to the user command interface.

4.7.4.3 EEI Services

On a traditional system, these capabilities are implemented by providing interactive commands entered via a keyboard. However, as graphical user interfaces evolve, these commands may also be implemented by clicking on a mouse with the cursor in a particular area of the screen, by a touch screen, a tablet, or other input device.

The major services at the EEI provide the following abilities:

- Capture the output of a command or application into a file
- Redirect the input for a command from a file
- Direct the output of a command to be used as the input to another command
- Execute applications
- Get online help for commands or applications
- Manipulate file contents
 - Cut
 - Paste
 - Concatenate
 - Convert
 - Sort
 - Reformat
 - Compare
 - Search for regular expression
- Edit files
 - Interactive editors
 - Batch or “stream” editors
- Display files
 - Pausing when necessary
 - Display only selected ranges of files
- Manipulate files
 - Create
 - Delete
 - Rename
 - Move
 - Copy
- Print files
- Perform network functions
 - Transfer files
 - Execute commands remotely
 - Print files remotely

- Perform batch processing
 - Create and manage batch queues
 - Submit, terminate, and get status of jobs
 - Retrieve output
- Manipulate and display directories
 - Create
 - Delete
 - Display
 - Destroy (delete a directory and all its subdirectories and files)
- Control file and directory permissions
- Communicate with other users
 - Send and receive electronic mail
 - Interact online (two or more users communicating with each other simultaneously)
- Interrogate system information
 - Currently logged on users
 - Other information about users
 - Currently executing system processes
 - Kernel, disk interface, and network status
- Control the application execution environment
 - Execute applications in the background
 - Abort applications running in the foreground or background
 - Suspend an application
 - Move an application running in foreground mode to the background
- Schedule commands for periodic execution
- Control the input equipment of the user, such as a terminal or graphical user interface
- Manage local environment and configuration information
- Query local environment and configuration data
- Configure an environment for an international locale

4.7.4.4 Interapplication Services

The following services enable remote users and applications to access and execute user command interfaces of a system as if they were directly connected to that system. The major categories of interapplication entity services include the following:

- Login and use hosts on a network as if the users logging in were directly connected to a local terminal
- Remotely execute the shell commands of a system as if the user were directly connected to a local terminal
- Copy files between hosts without going through a network file transfer program
- Find out who else is logged into the machines on a local-area network
- Query the status and uptime of all machines on a local-area network

4.7.5 Standards, Specifications, and Gaps

There is currently only one formal standard for user command interfaces: ISO/IEC 9945-2 :1993 {69}, which is identical in content to IEEE Std 1003.2-1992. Several other user-command interface standards-development activities are also underway. In addition, there are several consortia-defined specifications and de facto specification standards for shell and utilities services and interfaces. Table 4.15 lists the standards concerned with command interfaces.

Table 4.15—User Command Interface Processing Standards

Service	Type	Specification	Subclause
Shell and utilities	S	ISO/IEC 9945-2 : 1993 {69}	4.7.5.1
User portability utilities	S	ISO/IEC 9945-2 : 1993 {69} UPE	4.7.5.1
Security utilities	E	IEEE P1003.2c {B25}	4.7.5.2.1
Batch environment	S	IEEE Std 1003.2d-1994 {118}	4.7.5.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.16 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE. (The entry for ISO/IEC 9945-2 : 1993 {69} applies only to the API portion of that standard.)

Table 4.16—User Command Interface Processing Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
ISO/IEC 9945-2 : 1993 {69}	E				S		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
ISO/IEC 9945-2 : 1993 {69}						

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.7.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.7.5.1.1 POSIX Shell and Utilities

ISO/IEC 9945-2 :1993 {69} comprises two classes of service, developed originally as two separate documents (IEEE P1003.2 and P1003.2a):

- A command set that provides access to a specific set of services and utilities commonly used by people interacting with computer systems. These services and programs are complementary to those specified by ISO/IEC 9945-1 :1990 {68}.
- Additional, optional utilities and features that promote the portability of users and programmers, in addition to applications, across conforming systems—the User Portability Utilities Option. The consistent interactive environment does not include emerging technologies such as graphical user interfaces, which are under development by different standards groups.

Parts of ISO/IEC 9945-2 :1993 {69} go beyond the current services and include a number of software development and debugging utility services. These are addressed in 4.11 on application software development services.

4.7.5.1.2 Batch Environment

IEEE Std 1003.2d-1994 {118} extends ISO/IEC 9945-2 :1993 {69} to define utilities, system administration interfaces, and an application-level protocol that address the following areas:

- Utilities for submission and management of requests
- System administration interfaces for the creation, management, and authorization of the network queueing and batch processing system
- Application-level network protocols

There are no formal interapplication standards that address the remote access and execution of the command interfaces of a system. The 4.3BSD document {B54} addresses all these services, however.

IEEE Std 1003.2d-1994 {118} is based on the Network Queuing System (NQS), which is used in many large computing facilities.

4.7.5.2 Additional Specifications

4.7.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.7.5.2.1.1 IEEE P1003.2c

The IEEE P1003.6 working group is developing an amendment to ISO/IEC 9945-2 :1993 {69} to address security enhancement requirements:

- Clarification of ISO/IEC 9945-2 :1993 {69} utility behavior on systems with additional or alternate security mechanisms.
- Additional utilities for user and shell-application control of access control lists and other aspects of the security interface.

See 5.2.

4.7.5.2.2 Public Specifications

None.

4.7.5.3 Unaddressed Services

Unaddressed user command interface service areas include the traditional UNIX system-based command interfaces for remote command execution, remote file copy, remote login, gather remote host and user statistics (e.g., `uux`, `rsh`, `rcp`, `rlogin`, `rwho`, `ruptime`, and anonymous FTP). Specifications that address these areas are available from a number of organizations. These include

- OSF AES-OSC {B50}
- The X/Open XPG4 {B69} specifications (`uux` only)
- Internet suite applications [e.g., RFC-959 {B41} (FTP), RFC-854 {B40} (Telnet), RFC-821 {B38} (SMTP)]

4.7.6 POSIX OSE Cross-Category Services

4.7.6.1 Internationalization Services

The utilities described in ISO/IEC 9945-2 :1993 {69} satisfy some requirements for standardized multilingual and multicultural support (e.g., localization requirements such as date formats and collation sequences, and support for international character sets).

4.7.7 Related Standards

IEEE P1387.4 {B33} defines POSIX extensions for printing in a distributed (networked) environment. The POSIX Printing System command-line and programmatic interfaces are based on those in the Palladium Print System from MIT's Project Athena. The commands for remote printing are addressed in 5.3.

Many of the commands for software development services are discussed in 4.11.

4.8 Character-Based User Interface Services

4.8.1 Overview and Rationale

This clause describes the system services that are related to character-based terminals. It describes both the APIs to character-based terminals and also the look and feel of the interaction between the user and the user interface equipment.

Despite the attention paid to graphical window interfaces, a great number of applications still are written with a character-based user interface. Because character-based services are less demanding in term of resources (memory, processing, resolution of the display device), they are well suited for applications that do not require a sophisticated interaction with the user and when the cost of the display devices is a major concern.

Character-based user interfaces are characterized by the way they interact with the displaying device. They present information such as text exclusively by sending codes to the display device. They can activate the display only by these codes, which are constrained in a fixed box of dots (pixels), as opposed to bitmapped control of the display where each pixel is under control.

Because of this mode of interaction with the display device, character-based interfaces are rarely used to display precise graphical information, although there are some codesets called *alphamosaic*, which can be used to construct elaborate illustrations with coarse rendering.

The character-based user interfaces are also characterized by the ways they handle the interaction with the terminal. With *character-stream* terminals, the output and input operations are not necessarily structured. With *block-mode* terminals, the application displays a structured screen where the user fills all the desired input before the answers are sent back to the application.

It should be noted also that there are character-based window applications that need not have all the flexibility and ease of use of their graphic counterparts, but represent an alternative allowing the utilization of the large installed base of character terminals and still improve the ease of use. Forms-based interfaces are not necessarily dependent upon the terminal hardware and provide the user with a structured method to input data, which is combined into a data-format (form) for producing a completed formatted document as output. The character-based services do not include the services associated with the connection to a remote application platform by a mechanism called *virtual terminal* because the latter are considered more a communication service offered to a user than a specific API to an application.

4.8.2 Scope

The scope of this clause is limited to the services and standards that support character (nonbitmapped) terminals. The services described here do not preclude the use of block-mode terminals, even though most applications built on POSIX-conforming platforms historically have used character-stream terminals.

4.8.3 Reference Model

This subclause identifies the entities and interfaces specific to the character-based user interface services of an OSE.

As illustrated in Figure 4.11, the components of character-based interfaces are broken into two groups: those specifications that impact the API and those that impact the external user interface.

This reference model is consistent with, and expands on, the reference model in Section 3.

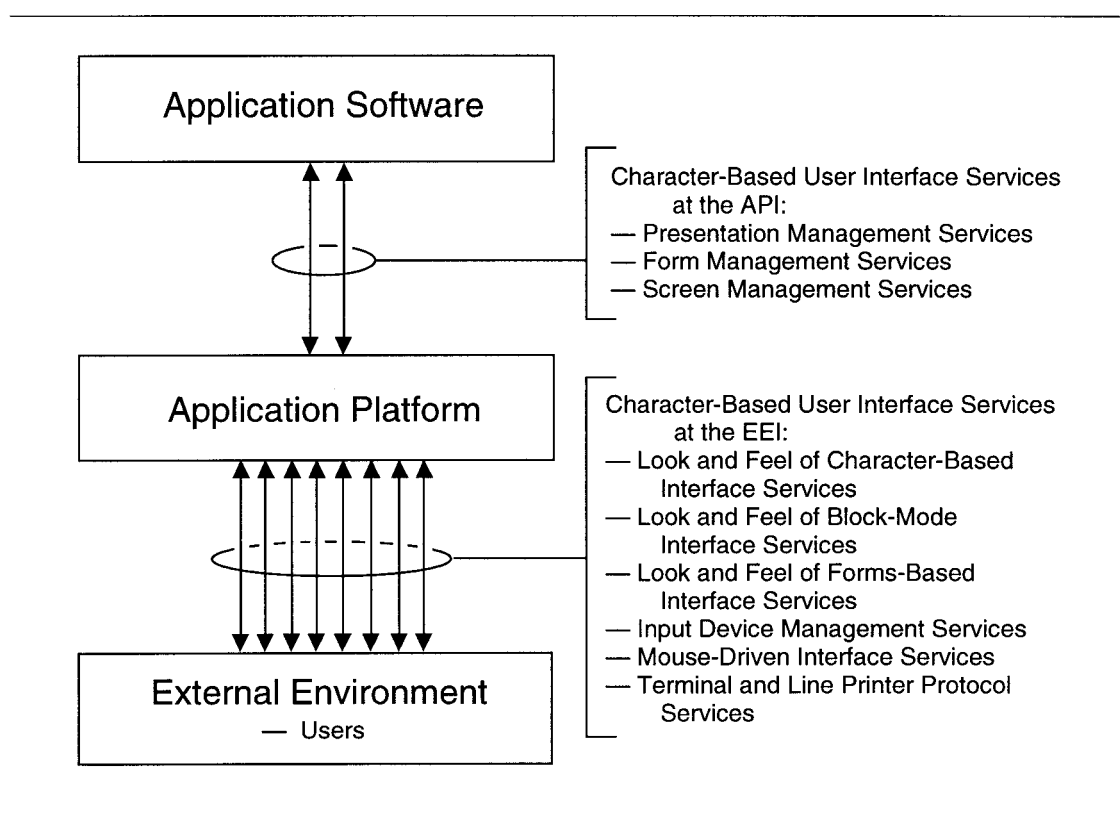


Figure 4.11—POSIX OSE Character-Based User Interface Services Reference Model

4.8.4 Services

The fundamental services for character-based terminals are to allow applications to be written that make use of the features of a wide variety of terminals using a single terminal-independent interface. The look and feel of user interactions should be consistent between applications for a particular user organization or enterprise in order to make moving between applications as simple as possible.

4.8.4.1 API Services

Application services include those made available to the application developer to separate the application function from the user interface functions as much as possible.

These standard services support application portability and terminal independence.

4.8.4.1.1 Presentation Management

Services available for presentation management are

- Placement of text on the screen using a consistent coordinate system reference
- Positioning of the cursor for further output on the screen or for user input
- Control of attributes of displayed text such as highlighting, underscoring, and coloring, if available
- Clearing or refreshing the screen
- Getting the current cursor position

4.8.4.1.2 Screen Management

Services available for screen management are

- Control of the number and the width of the lines displayed
- Use of a protected status line
- Protection from writing or clearing in defined portions of the screen
- Autowrapping in defined portions of the screen

4.8.4.1.3 Form Management

Services available for form management are

- Definition of a form with different output and input text fields
- Definition of the input field attributes, such as text or different numeric formats
- Generic and customizable error handling procedures for incorrect input

4.8.4.2 EEI Services

4.8.4.2.1 Look and Feel

The look and feel of user interactions with applications should be standardized to make moving between applications as simple as possible. The areas that require standardization are

- Style of selecting commands
- Accessing online help
- Performing common functions such as page forward and page backwards
- Selecting or moving between fields in a forms-based environment

These interactions may differ slightly between different types of terminals because of limitations of the terminals.

4.8.4.2.2 Keyboard Layout

The keyboard layout could be standardized. Even if the characters of different national languages force keyboards to have different layouts, it is desirable that the special keys, such as “escape,” “return,” “tab,” and the different function keys, have an unambiguous representation.

However, it is recognized that the actual layout of most popular keyboards are not optimum in term of ergonomics. Keyboard layout is often cited as the example of a de facto standard that was designed to account for the technical limitations of early typewriters, so care has to be taken in trying to enforce keyboard layout.

4.8.4.2.3 Mouse-Driven Interface

Character-mode user interfaces are not limited to keyboard-oriented I/O. Some character-mode user interfaces support mouse-driven I/O.

4.8.4.2.4 Terminal and Line Printer Protocols

The terminal and printer protocol definitions should be hidden from the applications. However, for the purpose of interoperability and management (initialization, diagnostics), these protocols may be useful. Virtual terminal definition falls into this category of services.

4.8.4.3 Related Services

A definition of character that could be recognized could be useful.

4.8.5 Standards, Specifications, and Gaps

See Table 4.17.

Table 4.17—Character-Based User Interface Standards

Service	Type	Specification	Subclause
FIMS	S	ISO/IEC 11730 : 1994 {92}	4.8.5.1
Terminal protocol	S	ISO/IEC 11730 : 1994 {92}	4.8.5.1
Keyboard layout	S	ISO/IEC 9995 : 1994 {70}	4.8.5.1
OCR	S	ANSI X3.62-1987 {104}	4.8.5.1

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.8.5.1 Standards in the POSIX OSE

4.8.5.1.1 Forms Interface Management System (FIMS)

ISO/IEC11730 : 1994 {92} is based on the CODASYL document. It addresses some of the services for a forms-based user interface.

4.8.5.2 Additional Specifications

4.8.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations.

Table 4.18—Character-Based User Interface Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
ISO/IEC11730 : 1994 {92}	S						

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
ISO/IEC11730 : 1994 {92}						

NOTES:

- 1 — LIS—Language-independent specification is available.
- 2 — Ada, APL, Basic,... —Language-dependent specification is available.
- 3 — S, E, P, G—Standard, Emerging Standard, Public Specification, Gap

Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.8.5.2.2 IEEE P1201.1 {B30} Windowing API

The IEEE P1201.1 working group is developing a windowing API that is based on the XVT Portability toolkit. This toolkit supports a mouse-driven character-mode user interface.

4.8.5.2.3 Public Specifications

None.

4.8.5.3 Unaddressed Services

- Presentation management services (X/Open has included the “curses” library in X/Open XPG4 {B69})
- Screen management services
- At the API:
 - Presentation management
- At the EEI:
 - Look and feel
 - Screen management
 - Input device management
 - Line printer protocols

4.8.6 POSIX OSE Cross-Category Services**4.8.6.1 Security**

The security aspects of the interaction with a character-based terminal are related to the assurance for the user that he or she is really interacting with the right application and not a false one. This is done in some systems by the use of a special sequence or combination of characters that provide assurance that the user has accessed a *trusted* application. This mechanism is called *trusted path*.

Another mechanism is the encryption of the character exchange between the terminal and the application platform. In some systems this could be done by a personal device in some specialized “smart” cards.

4.8.6.2 System Management

It is important to allow system management personnel to configure the system to designate where each terminal is connected. Also needed is the ability to add support for new terminals without affecting the application interface.

4.8.6.2.1 Configuration Management

The system could include a descriptive database of a current set of supported terminals so that terminal-independent services can do the mapping for the different functions. These databases are present in some POSIX-conforming platforms. They are known as *termcap* or *terminfo*, depending on the implementation. They should not be referenced directly by an application. The way they are specified or formally described is not yet officially standardized because they deal more with a PII that is out of the scope of application portability.

4.8.7 Related Standards

None.

4.9 Windowing System Services

4.9.1 Overview and Rationale

The windowing system services are a modern facility of computer systems that support direct user-machine interaction. Until recently, most computer operating systems interpreted commands that were typed in from the keyboard of an alphanumeric computer terminal. Special-purpose applications, such as those for CAD/CAM, have always presented user interfaces based on a series of menus or by pointing at visual displays with tablets and lightpens. The availability of low-cost bitmapped graphic workstations and personal computers has led to the proliferation of graphical user interfaces (GUIs), windowing technologies, generic commands (e.g., File, Save, Save As, New, etc.), and an assortment of selection techniques via mouse, track ball, tablet, etc. In several of these technologies, de facto standards are emerging and becoming informally accepted by the user community, and with more frequency, mandated for use in systems being developed within government agencies and private industry. The primary motivations for considering graphical window system standards and their relation to POSIX standards include

- The existence and popularity of windowing systems
- The requirement for development of applications that take advantage of the windowing system environment
- The requirement of many users and manufacturers for a basic consistency in the presentation and behavior of graphical window systems across multiple graphics platforms

As the windowing system technology evolves within the graphics environment, the differences between windowing services and graphic services becomes less distinct. The distinction for purposes of this document is that graphic services are associated with providing general-purpose interfaces for creating virtually any kind of two- and three-dimensional graphics (e.g., GKS for two-dimensional graphics and PHIGS for three-dimensional graphics). Graphical window system services certainly utilize graphic technologies, but they are limited to providing graphics related to window-based user interfaces and specifications on how users may interact with an application within a window environment. The graphic services are addressed independently in 4.10.

4.9.2 Scope

Standards and standards initiatives concerning the subject of graphical window system interfaces cover a wide area ranging from keyboard layout to screen management. In this clause, the following specific standards are considered:

- Protocols for window management on a local or remote display device
- APIs for such protocols

- Graphical window system drivability features that define a common subset of “look and feel”; i.e., appearance, screen positioning, and behavior of graphical window system objects within windows on a graphic screen
- API specifications for the appropriate associated language bindings and the display, manipulation, and management of interaction objects within windows on a graphic screen
- Command-language interfaces that may be entered interactively by the user or retrieved from a stored procedure
- API specifications and appropriate associated language bindings required to support character (nonbitmapped) terminals
- API specifications and appropriate associated language bindings for the translation, manipulation, and management of command statements (or messages)

Standards relating to the following are addressed in other clauses:

- Graphics; see 4.10
- Network transport protocols; see 4.3

4.9.3 Reference Model

This subclause identifies the entities and interfaces specific to the construction of an application using a graphical window system. As illustrated in Figure 4.12, the interface components involved in the user interface process are divided into the EEI and the API.

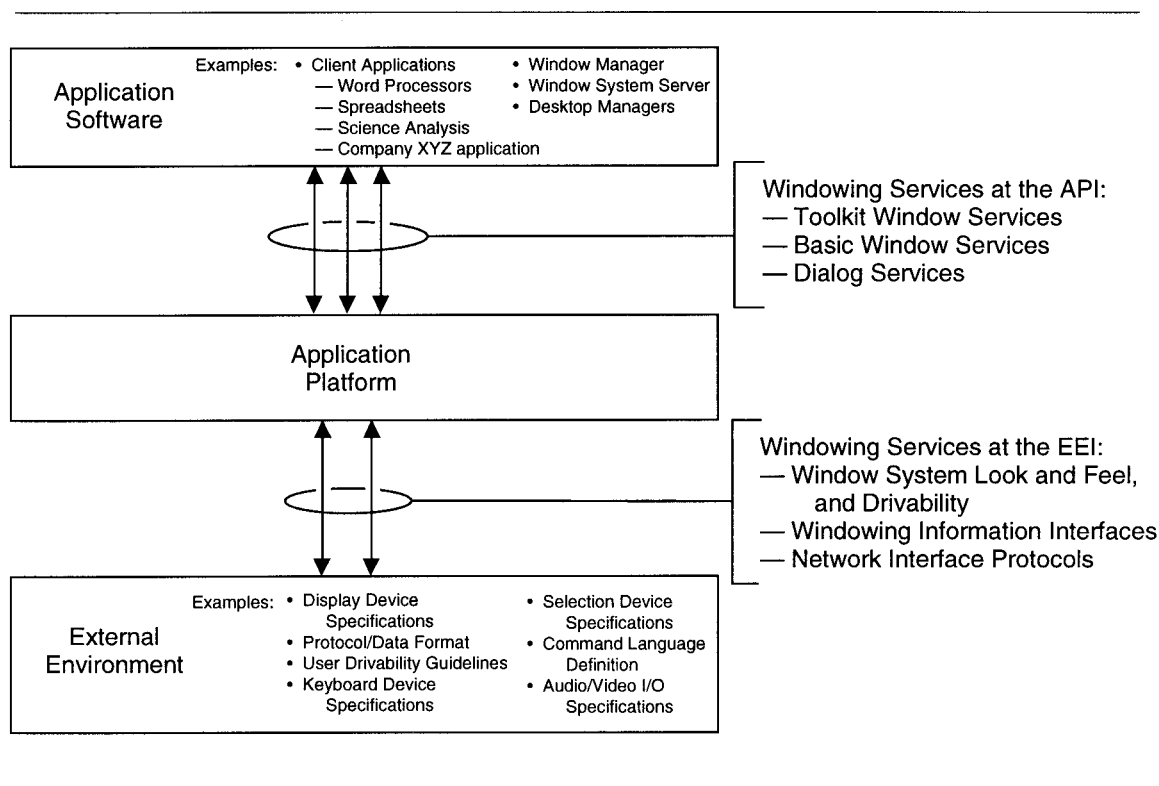


Figure 4.12—POSIX OSE Windowing System Services Reference Model

The EEI is concerned with the communication with the user via the physical graphical window system devices (e.g., keyboard, terminal, mouse, display screen). The applicable EEI standards are driven primarily in support of user and

data portability across different application platforms. Current efforts within the IEEE P1201.2 working group are intended to define a minimal set of commonality in graphical window systems, which will eliminate problem areas such as

- Error provoking inconsistencies
- Misleading expectations about the results of user actions
- Gross inconsistencies in the high-level user model or metaphor
- Unnecessarily difficult or numerous hand motions

The drivability concept derives its name from the concept of “driving” an interface. A frequently cited analogy is the automobile. Having a standard location for the clutch, brake, accelerator pedals, ignition key, and steering wheel allow drivers to move between car models with relative ease (until they have to roll down the window or turn on the lights or windshield wipers). Similarly, the EEI drivability guidelines will provide standards for graphical window systems that will ensure ease of moving between application platform models. For example, which mouse click causes an interaction object (e.g., radio button) to be selected or how a scroll bar should behave would be candidates for standard EEI specification.

The API is concerned with the interface between the application semantics and the graphical window system services. It is the interface between the application software and the application platform and is defined primarily in support of application portability. These services provide functions for creation and manipulation of visual display objects such as menus, buttons, scrollbars, and dialogue boxes. In addition, these functions allow information about user actions to flow back to the application software; for example, when the user has selected an item from a menu. This information about user actions is known as an event. Applications that require communication with the user are inherently event-driven. That is, associated with dialogue window of an application (i.e., a window in which a user response is expected) is a main event loop waiting for the user to make a selection that will trigger an operation to be performed by the application.

The API supports an abstract view of the windowing user interface, specified in terms of visual objects, their attributes, and relationships among them. The visual appearance of two different implementations may be totally different while presenting the same services to the application. The API definition ensures that the application software can be ported across POSIX-compliant platforms; and the API supports the EEI drivability guidelines, enabling users to operate the application across platforms easily.

4.9.3.1 Application Software Elements

Application software elements include

Window system server

The window system server provides a function that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Applications and other programs that use basic windowing services are called “clients.” Many clients may talk to the same server. The server is independent of operating system, programming languages, or network communication.

Window manager

A Window Manager provides a uniform method for manipulating windows, which includes a basic set of window management capabilities that allow for development of alternative or user-preferred window managers or both, through communication of the Window Manager with the Windows System Server. Window Manager capabilities include, but are not limited to

- Resize window
- Move window
- Push/pop window to top/bottom
- Shrink window to a reduced visual representation of window (i.e., frequently referred to as an icon of the window)
- Audio I/O

- Video I/O
- Security

Desktop

The desktop, or graphical windowing shell, is a specification for the graphical window system work surface (i.e., the entire display screen).

The desktop provides the user with a visual interface to available computer resources. A desktop may be characterized as a visual analog of the POSIX shell. It provides access to system resources, such as devices and files, and provides methods to start applications. Desktops typically also provide a set of often-used utilities such as a calendar, a notepad, etc. Desktops also typically provide the capabilities for end users to customize their work desktop environments easily, based on their own preferences or needs. The desktop is an important component of the look and feel of a graphical window system, but the current state of the industry is too immature for any standardization to materialize on a desktop specification in the immediate future.

Local and remote applications

These applications are clients that provide the functions required to perform the specific task(s) that the user needs to achieve (e.g., spreadsheets, scientific analysis systems, CASE tools, and process and control tasks.)

4.9.4 Services

Windowing system services provide a controlled interface between the application-specific software and the user-interface-specific software, allowing each to be designed and implemented separately. Users of these services include all POSIX system users and those charged with maintaining the processor and graphical window system communication. A common, standardized graphical window system for applications should be available to users across the POSIX OSE.

4.9.4.1 API Services

Windowing system services include those services made available to the application developer to separate the application functions from the graphical window system functions as much as possible. They include such areas as screen management, windowing, and user input device services.

These standard services support requirements for application portability, software commonality, application interoperability, and data communication transparency.

A programmer may access the following services for an application via language bindings.

4.9.4.1.1 Basic Window Services

The basic window services, which can be called from client applications, support a window-based user interface. They should be based on a “client-server” model. The server is a program that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Many clients may talk to the same server.

The major functional areas are

- Window management
- Desktop management
- Event handling
- Error handling
- Interclient communications
- Input device management: keyboard, pointing device
- Screen management
- User preferences management
- Server connection management

The following functions are available under each functional area.

4.9.4.1.1.1 Window Management

Functions available for window management are

- Create a window, map a window onto the screen, delete a window (includes support for character-based emulator window)
- Manipulate a window (move, resize, change view precedence)
- Manipulate window attributes (set, get, change; attributes may be related to appearance, redraw performance, event handling, or change authority)
- Seize and relinquish control over the server for display purposes (permits uninterrupted client output; output requests from other clients will be queued and displayed later)

4.9.4.1.1.2 Desktop Management

Functions available for desktop management are

- Associate data with a window (e.g., context manager functions, association table functions, etc.)
- Manipulate the graphics context for a given object (create a graphics context, obtain current graphics context, change graphics context)
- Get and set fonts (load font, list fonts, unload font, obtain font metric information)
- Draw graphics primitives (draw arc, draw line, fill polygon, clear rectangular window, clear entire window)
- Manipulate window cursors (create, destroy, assign, change)
- Draw text and obtain text metric information

4.9.4.1.1.3 Event Handling

The basic window services require applications to respond to the actions of the user, rather than forcing the user to respond to the application in a rigid, serialized manner. This requirement necessitates that a program either

- 1) Be capable of handling any one of a number of events at any single point in time, or
- 2) Associate a routine with each event to be called automatically when that event occurs

There is a separate set of events for each window used by the application. An application selects the events for a particular window, maps the selected events to the window, and reads events from the event queue as they occur. There are three major types of events:

- Input device events (button press event, button release event, key press event, key release event)
- Window management events (e.g., window resize)
- Client message events [selection data transferred (by another application) event, private interclient communication event]

Functions available for event handling are

- Select events
- Map events to a window
- Get information about events
- Send events

4.9.4.1.1.4 Error Handling

Functions available for error handling are

- Get error message
- Get error description
- Set error event handler routine

4.9.4.1.1.5 Interclient Communication

The basic window services are required to be network transparent to an application or client. This means that an application on one host may write to the display screen connected to another host without being aware that networking is involved. The basic window services handle the network connections and follow the protocols necessary for the application to interact with the display. This convention allows redistribution of applications in a networked system with no effect on the application software. Therefore, an application client cannot assume that another client can open the same files or seize the same processing environment. Interclient communication via the server has two forms:

Properties

Clients may associate arbitrary information with a window; generally used for communication between a client and the window manager.

Selections

Selections are selected by the user out of a window and can be “sent” to another window (even if it belongs to another client). The receiver of the selection can decide the subsequent action, such as displaying the selected data in its window.

Functions available for interclient communication include

- Manipulate window properties (list, delete, change, get)
- Set and get selections
- Manipulate cut buffers

4.9.4.1.1.6 Input Device Management

Functions available for input device management include

- Receive keyboard input and pointing device button events
- Gain exclusive control of keyboard or pointing cursor
- Track the pointing cursor
- Change server-wide keyboard mappings
- Set and get keyboard and pointing device preferences

4.9.4.1.1.7 Screen Management

Functions available for screen management are

- Manipulate color using colormaps (copy, change, install, deinstall, get default)
- Get, display, and manipulate bitmapped screen images
- Screen saver functions (blanking screen on idle)
- Retrieve display information (default colormap, number of display planes, screen width, and height)

4.9.4.1.1.8 User Preferences Management

The services and data structures used for managing user preferences are provided and collectively referred to as user preferences management. User preferences management may include the following options that need to be read and merged:

- The defaults of the user stored in the user resource manager property of the root window
- The defaults of the user stored in a defaults file of the user

- The defaults of the application program
- The options specified at application startup (e.g., command-line arguments)

Functions available for user preferences management are

- Set and get preference data

4.9.4.1.1.9 Server Connection Management

Functions available for server connection management are

- Control access to the server [add host to the access control list (ACL), list ACL, disable ACL]
- Connect and disconnect a client from a server (and the display controlled by the server)
- Obtain server implementation information
- Flush output buffer to the server and wait for the server to process all requests in the output buffer

4.9.4.1.2 Toolkit Window Services

The toolkit window services provide a mechanism for runtime access to a library of visual objects. A visual object is a graphical display object (i.e., interaction object) with associated software that receives input from users (typically via a keyboard and a pointing device) and communicates with applications and other visual object software. The graphical representation of a visual object can be modified to reflect the results of application processing. Examples of visual objects¹⁸ are graphical push buttons, check boxes, and editing boxes.

Toolkit window services are provided for two reasons:

- To allow application software to utilize a visual object library directly
 - To allow application-specific visual objects to be created and added to the widget library
- NOTE — Creating a visual object includes writing software that uses the X Toolkit services.

Therefore, toolkit services may be logically divided into two categories, with some overlap: visual object interface services, which are called by an application or dialogue service, and visual object programming services, which are called by the visual object software.

An application may use toolkit window services to

- Perform toolkit initialization/exit
- Set up visual object resources
- Create/delete a visual object
- Display a visual object
- Add/remove application-specific routines to be called by a visual object (event callbacks)
- Retrieve/modify the state of a visual object
- Turn control over to the toolkit for user input processing

A visual object software program may use toolkit window services to

- Manage child visual objects (a child visual object is a visual object that is displayed inside of another visual object)
- Manage window events, timer events, and file input events
- Handle visual object geometry (sizing, positioning, child visual object placement)
- Handle user input
- Manage visual object resources

¹⁸ The term used within the X Toolkit [B46] to define visual objects is “widgets.”

- Translate an event into an action
- Manipulate graphics contexts
- Manipulate pixmaps (pixel arrays—used to display a graphical object by turning pixels on and off)
- Manage memory associated with graphical window systems
- Handle errors associated with graphical window systems
- Allow intervisual object communication (via the selection mechanism)
- Initiate other visual object routines (visual object event callbacks)
- Initiate application-specific routines that have been associated with the visual object by the application (application event callbacks)

4.9.4.1.3 Dialogue Services

Dialogue services provide functions to support high-level graphical window system management for applications, with the primary goal of delivering user inputs to the application program and application-driven information to the user. The dialogue services allow for a separation of the user interface specifications from the application program. For example, there are many applications that are not concerned with whether a user entry object is a pull-down menu or a scrollable list. These applications are only interested in what the user specified or selected from the user entry object (i.e., the parameter value), which will then trigger some action by the application. To support this notion, a single dialogue function might be specified for displaying a window made up of a composite of visual display objects, such as radio buttons, text key-in objects, and scrollable text lists. The application program does not need to manage or understand what the look, location, or visual feedback of any of these items will be. The dialogue function has access to the presentation information required to display the specified window and it handles the display of the application-specified window. Another dialogue service would provide a high-level event loop that returns the user-specified input as an application parameter value.

These services provide simplicity over the degree of freedom available in basic and toolkit window services. Most User Interface Management Systems (UIMs) provide dialogue services to fulfill their requirement of separation of user interface from application software.

These services are subdivided into

Window services

Provide services used to initialize the window service, create and delete windows with predefined associated visual objects, and manipulate the pointing cursor. Includes services that allow the application to communicate directly with the user via modal or modeless windows.

Visual object manipulation services

Provide services used to access the graphical window system designed by the application designer, display the visual objects defined by the graphical window system, and associate them with application-tied inputs and outputs.

Event control services

Provide services to allow the application to define a set of events and handle triggered events in one of two ways:

- Wait on the occurrence of any event, processing triggered events one at a time from an input queue (event-driven method)
- Attach to each event a function that is automatically executed when the event is triggered (callback method)

4.9.4.2 EEI Services

These services provide support for the actual elements with which the user physically interacts. These functions provide services in three areas:

Window system look and feel, and drivability

Provides definition of the presentation and behavior of the visual display objects; command language definition (syntax and semantics); and specifications related to keyboards, selection devices, and audio and video input/output devices.

Windowing information interfaces

Provides specification of user resource data formats, containing presentation and action information pertaining to visual display objects.

Network interface protocols

Provides protocol services for data transport, which are basically the bottom six layers of the OSI model.

4.9.4.3 Interapplication Entity Communication Services

These services provide generalized communication support for conventions and specifications for interaction between graphical window system components. The services provide support for generalized network/multisession services, such as message handling between graphical window system components, intermediate language definition, and a standard definition of the format used for saving the presentation, behavior, and action information about graphical window system objects. These services provide a generalized communication protocol that is necessary for exchange of information between windowing systems.

4.9.4.4 Windowing Resource Management Services

These services provide general management functions across the graphical window system components, which include system-administration-oriented functions. (That is, management of graphical window systems within the scope of the administrator, such as setting up defaults and user customization functions. For instance, it is important to allow reconfiguration and addition of terminals and displays without affecting the application interface.) These resource management services are independent from the OLTP Resource Management Services defined in 4.6.4.3.

A standard definition of the format used for saving the presentation, behavior, and action information about graphical window system objects would provide a vehicle for exchanging graphical window system information between software tools, such as UIMSs and Interface Design Tool (IDTs).

4.9.5 Standards, Specifications, and Gaps

See Table 4.19.

Table 4.20 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

4.9.5.1 Standards in the POSIX OSE

4.9.5.1.1 IEEE Std 1295-1993 {127}

IEEE Std 1295-1993 {127} is a standard based on the X Window System intrinsics and Xlib. It provides definitions for the toolkit layer of the X Window System, based on OSF/Motif specifications.

Table 4.19—Windowing Standards

Service	Type	Specification	Subclause
Basic window	P	X Window System Xlib {B52}	4.9.5.2.2
Toolkit	S	IEEE Std 1295-1993 {127}	4.9.5.1
Dialogue	E	IEEE P1201.1 {B30}	4.9.5.2.1
	P	X Window System X Toolkit {B46}	4.9.5.2.2
	E	IEEE P1201.2 {B31}	4.9.5.2.1
User-System interfaces and symbols	S	ISO/IEC 9995 : 1994 {70}	4.9.5.1
Presentation, form-based dialogue, and window-based interaction	S	ISO/IEC 11730 : 1994 {92}	4.9.5.1
Software engineering and man-machine dialogue	S	ISO 9241 {49}	4.9.5.1
Uniform APIs	E	IEEE P1201.1 {B30}	4.9.5.2.1
Recommended guidelines for graphical user interface drivability	E	IEEE P1201.2 {B31}	4.9.5.2.1
Interapplication entity	P	X Window System (X Protocol) {B52}	4.9.5.2.2
Window/Character resource management	G	n/a	4.9.5.3

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.9.5.1.2 FIMS

The CODASYL committee has developed ISO/IEC 11730 :1994 {92} for FIMS, which covers the interface between a programming language and any form fill-in application on a computer or terminal screen.

4.9.5.1.3 ISO/IEC 9995: 1994 {70}

This standard was developed by the User-System Interfaces and Symbols committee. The scope of the standard is keyboard layouts for text and office systems.

4.9.5.1.4 ISO 9241 {49}

This standard covers various aspects of the ergonomic requirements for office work with visual display terminals (VDTs), including keyboard, visual, workplace, color, and usability requirements.

4.9.5.2 Additional Specifications

4.9.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

Table 4.20—Windowing Graphical Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
PHIGS					E		
GKS					E		
GKS-3D					E		
CGI					G		
IEEE P1201.1 {B30}					G		
IEEE Std 1295-1993 {127}					S		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
PHIGS						
GKS						
GKS-3D						
CGI						
IEEE P1201.1 {B30}						
IEEE Std 1295-1993 {127}						

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.9.5.2.1.1 ANSI HFS-HCI

This ANSI committee is working on drafts for the design process, information presentation, forms-based dialogues, and window-based interaction.

4.9.5.2.1.2 ISO TC159/SC4/WG5

The areas of concentration for this committee are software ergonomics, dialogue principles, dialogue styles, methods for evaluating software usability, coding and formatting of information, and terminology.

4.9.5.2.1.3 IEEE P1201.1

The IEEE P1201.1 {B30} working group is chartered to develop a standard that defines a uniform API that can be used by implementors of GUI applications to develop computer programs that are portable across multiple window systems and user interface toolkits, such as OSF/Motif, Microsoft Windows, OS/2 Presentation Manager, and Apple Macintosh.

4.9.5.2.1.4 IEEE P1201.2

The IEEE P1201.2 {B31} working group is chartered to develop a set of recommendations for graphical user interface drivability. The group is concerned with those elements and characteristics of HCIs that need to be consistent to permit users to transfer easily from one look and feel or application to another with minimal interference, errors, confusion,

relearning, and retraining. Topics covered include keyboards, pointing devices, menus, buttons, controls, windows, user guidance, and common user actions.¹⁹

4.9.5.2.2 Public Specifications

The specifications listed in this subclause are not part of the POSIX OSE, but may be of interest. Use of these specifications should be considered carefully because there are risks in using these unapproved specifications. They are included in this guide to indicate some of the existing work that has been done in areas that are gaps in the POSIX OSE.

There is a de facto standard for the base window system. The X Window System windowing protocol and the Xlib functional interface to the protocol were developed at MIT. Development is continuing under the aegis of the X Consortium, a group of interested parties in the computer industry and computer manufacturers. Relevant documents from the X Consortium are the Xlib and X Protocol specification {B52} and the X Toolkit specification {B46}.

The X Window System is a network-based windowing and two-dimensional graphics system. It uses the client-server model. The client and server can reside on the same or different platforms. The client is an application program executing anywhere on the network and displaying on the screen. It does this by making calls to a library called Xlib to generate protocols. The X server is the software that accepts X Protocol messages sent by the client and processes them for display. It also accepts input from a mouse or keyboard for return to the application program. The X Protocol specifies the data stream encoding between the server and the clients. The encoding will provide a standard interface for applications running on both distributed and nondistributed environments having high-speed, reliable, network-based communications.

X Protocol is designed to work in a heterogeneous network environment. Below the X Protocol, any lower layer of network can be used, as long as it is bidirectional. Currently TCP/IP and DECnet are the two network protocols commonly supported in X servers. Part 4 of this standard specifies the mapping of the X Window System protocol onto the OSI services.

4.9.5.2.2.1 Xlib

The Xlib—C-Language X Interface {B52} is the common component of the X Window System and resides on all X-based systems. Although X is fundamentally defined by a network protocol, application programmers do not interface directly with the X Protocol. Instead, they interface to the X Protocol through Xlib.

The X Window System uses the client-server model. The client is an application program executing anywhere on the network and displaying on the screen. It does this by making calls to Xlib to generate protocols. The X server is the software that accepts protocols sent by the client and processes them for display.

From a graphics perspective, Xlib is a two-dimensional graphics library and provides graphics primitives like points, lines, and arcs. It has a Graphics Context (GC) to allow modification of graphics attributes such as line type, line width, color, and font type. The Xlib developed initially at MIT is freely available and is a de facto standard for windowing and two-dimensional graphics. It has been adopted by major computer vendors and industry groups.

The X Window System protocol and functional interface are considered to be de facto standards in the base window system area because of their widespread adoption by major computer vendors and industry groups.

X/Open has published specifications for window management in conjunction with the X Consortium. *X Window System Protocol* {B56} contains a description and definition of the X Protocol. *X Window System File Formats and Application Conventions* {B58} describes formats and conventions for application cooperation and communication. *X Toolkit Intrinsics* {B57} contains a description of the X Toolkit functions and their use. *Xlib—C Language Binding* {B55} defines a programming interface to the X Window System protocol.

¹⁹The authorization for this project was withdrawn by the IEEE Standards Board in September 1995.

Within the United States government, NIST has adopted the X Window System Version 11 Release 3 X Window System protocol, Xlib, Xt Intrinsics, and Bitmap Distribution Format as FIPS Publication 158 {B19} . This is a noncompulsory (i.e., voluntary) standard.

4.9.5.3 Unaddressed Services

- Object definition file format: There are no standards addressing the format used for describing the “look and feel” of graphical window system objects
- Toolkit services
- Dialogue services
- Interapplication entity services

4.9.6 POSIX OSE Cross-Category Services

4.9.6.1 Security

The security aspects of graphical window systems include

- Option selections available in support of sensitive activities
- Authentication of a person at login
- Provisions for visual labeling of sensitive material
- Prevention of moving data (cut/paste) from a more protected security environment to a less protected environment

4.9.7 Related Standards

Currently, the basic windowing services provide a certain level of graphics functionality, but the existing and proposed graphics standards (e.g., PHIGS, GKS) provide a much more comprehensive solution to graphic support. As graphics and windowing technologies evolve, this distinction between windowing and graphics services will continue to be blurred. For instance, proposals are already being developed that provide extensions to the X Window System that support three-dimensional graphics (i.e., PEX, PHIGS EXtensions), and implementations of GKS are currently available that use the X Window System to create the graphics.

4.10 Graphics Services

4.10.1 Overview and Rationale

Graphics services are key components and play an important role in the POSIX OSE as it is used today in many different areas of industry, business, government, education, entertainment, and most recently, the home. The number of applications is growing rapidly, with increasing graphics capabilities. Some of these areas are user interfaces, computer-aided drafting and design, electronic publishing, plotting, simulation, animation, scientific visualization, art, and process control. The use of pictorial graphics provides a more intuitive interface and thus facilitates man/machine interaction.

Graphics have become a routine part of most organizations today, ranging from hardcopy graphs and charts to user interfaces and complex three-dimensional visualizations incorporating video and sound. The graphics technology of rendering objects has become dramatically more realistic and, hence, is used by engineers, architects, artists, etc., to enable them to see precisely what their final products, whether automobiles or buildings, will look like and behave under actual conditions.

Graphics have allowed dramatic improvements in the “look and feel” of user interfaces, and the trend is towards increasing use of these interfaces to interact with computers graphically, via windows and icons. This reduces the time involved in learning to use a computer.

Standardization of graphics services has many benefits for application developers, users, and systems integrators. The underlying motivations for considering graphics standards and their relation to the POSIX OSE include

Portability

Portability at both hardware and software levels is necessary to protect investment and achieve independence from a particular technology and a particular supplier of technology. There are many aspects of portability within graphics, all of which are potential money and time savers.

- Applications portability
- Graphics package portability
- Host machine independence
- Device independence
 - Input devices: dials, mouse, tablets, etc.
 - Output devices: plotters, raster, vector, etc.
- Window system independence
- Programming language independence
- Programmer portability
- User portability

Interoperability/Distributed graphics

Standard graphics protocols are necessary to allow applications to execute on one machine and display graphics on remote display servers. This allows for display of graphics on machines that are incapable of executing particular types of applications, and it also facilitates graphics conferencing.

Graphics data exchange

Standard graphics data exchange mechanisms are necessary to share or exchange graphical information between diverse applications.

This clause presents a reference model for this component and describes the services provided to application programmers and users. It also describes the current national and international standards, emerging standards, de facto standards, and any existing gaps that need new standardization efforts.

4.10.2 Scope

Included within this component are standards in the graphics area that address the following topics:

- API standards
- Language-binding standards
- Metafile and archive standards
- Device-independent interface and protocol standards
- Computer graphics reference model
- Conformance testing of implementations of graphics standards
- Distributed graphics standards
- Imaging standards
- Performance metrics standards

The standards not addressed here are

- Data exchange standards
- Graphical user interface standards
- Window management system standards

4.10.3 Reference Model

Over the past decade, many computer graphics standards have been developed. While they are similar in concepts, their underlying reference models are different. This restricts the degree to which the standards are compatible. By

producing a reference model to which all future graphics standards are to adhere, compatibility of graphics standards is maximized.

The ISO/IEC 11072 :1992 {91} Computer Graphics Reference Model (CGRM) standard defines a structure within which current and future international standards for computer graphics may be compared and their relationships described. It defines computer graphics in terms of five abstract levels called environments: construction, virtual, viewing, logical, and realization. Operations on data elements are defined in each environment, with computer graphics output expressed in terms of output primitives that make up a composition that is presented to the operator. Computer graphics input is expressed in terms of input tokens accumulated in an appropriate form for the application. Figure 4.13 presents the CGRM with its environments and external interfaces.

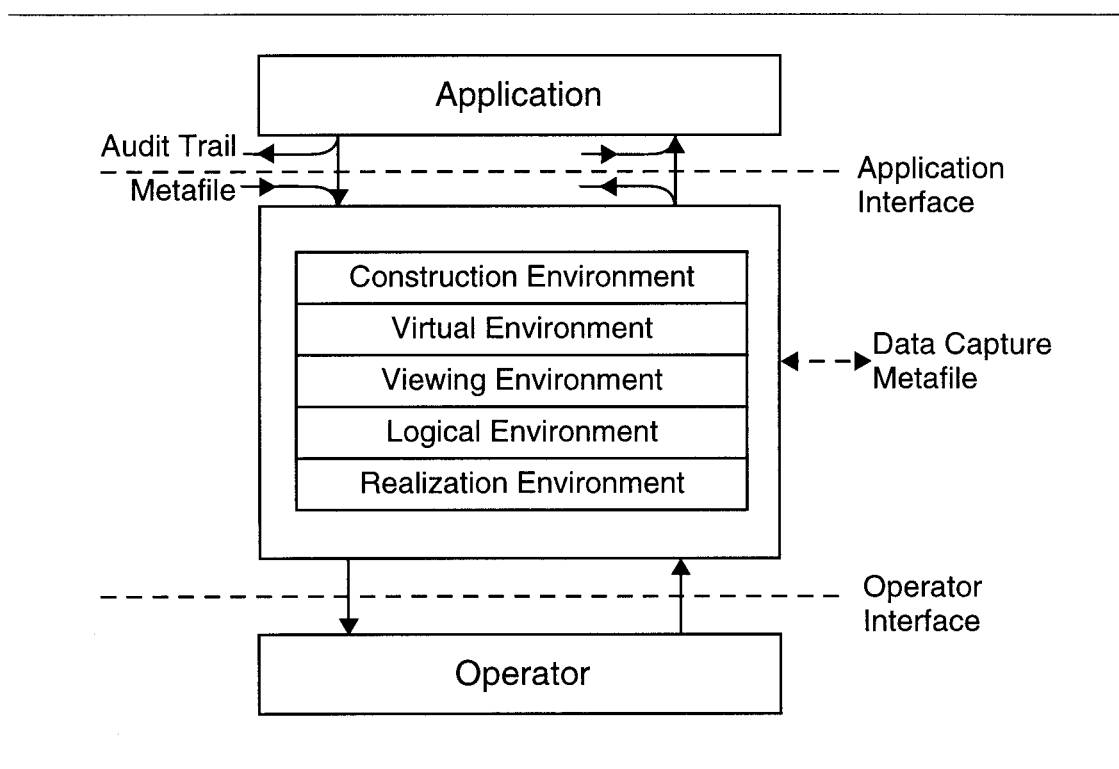


Figure 4.13—Computer Graphics Reference Model Level Structure

The application interface is the interface provided by the construction environment as the only interface between any computer graphics environment and the application. An audit trail metafile interface allows for record or playback of information flow across the application interface, and a data capture metafile interface allows for the import or export of all or part of the data elements. The operator interface is provided by the realization environment as the only interface between any computer graphics environment and the operator. The operator is the external object that observes the contents of the display in the realization environment and provides physical input tokens.

The CGRM can be incorporated into the POSIX OSE reference model as depicted in Figure 4.14. It provides the context for the discussion of graphics services and shows that the graphics services are a component of the overall POSIX OSE and is available to the application through the POSIX OSE API.

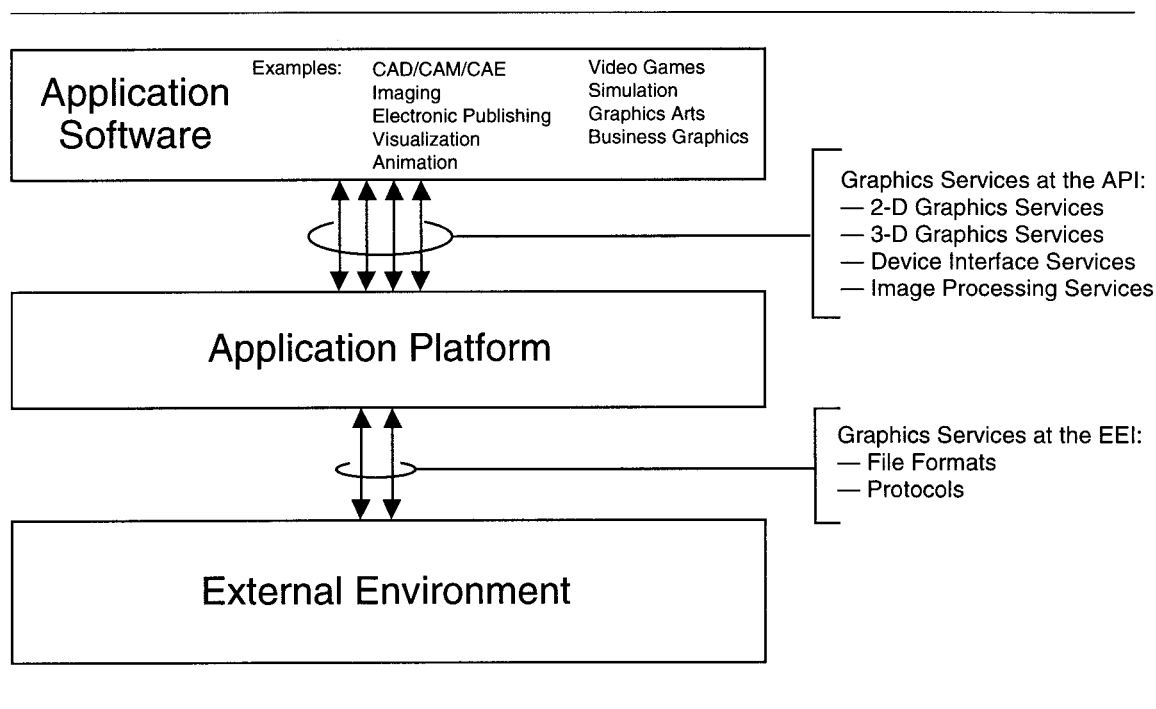


Figure 4.14—POSIX OSE Graphics Services Reference Model

The entities and interfaces specific to the graphics services are identified in this clause. The entities are

Application software

Such as CAD/CAM/CAE applications, imaging applications, and electronic publishing

Application platform

Representing the computer graphics environments that are realized by graphics libraries such as GKS and PHIGS

External environment

Consisting of external entities with which the application platform exchanges information, such as human users (or operators) and information interchange entities, identified as data capture metafile and audit trail metafile objects within the CGRM context

The interfaces are

- API** The API standardizes the conceptual model, calling sequence, functions, and syntax that a programmer uses to develop a graphics application. Each API standard has an attached language-binding standard that allows the functionality to be accessed from a variety of programming languages. A standard API in conjunction with a standard language binding promotes application portability by isolating the programmer from most hardware peculiarities and providing language features readily implemented on a broad range of processors. Examples of APIs in the graphics services area are GKS, PHIGS, PIK, etc.
- EEI** In the graphics services area, system software can be used for communication between the device-independent and the device-dependent functions, as well as protocols and file formats.

NOTE — The term HCI is used in this clause to be equivalent to the term “operator interface” of the CGRM.

The standardization efforts in the graphics area focus on these two interfaces.

4.10.4 Services

4.10.4.1 Graphics Concepts

Graphics services can be described in terms of the fundamental graphics concepts discussed in the following subclauses.

4.10.4.1.1 Graphical Primitives

The graphical primitives are the building blocks used to construct objects for display or storage in an archive file. Common graphical primitives are divided into classes of elements including: line elements, marker elements, text elements, filled-area elements, cell elements, and symbol elements. Each class of element has its own particular graphical primitives. For example, the line element primitives include polylines, arcs, and splines. Additionally, a generalized drawing primitive is a graphical primitive element that may be used to access device-specific graphical primitives.

4.10.4.1.2 Primitive Attributes

Primitive attributes determine the appearance of the graphical primitives and are classed as individual attributes or bundled attributes. Each of the classes of elements has its particular attributes, which may be bundled or individual. For example, a bundled set of line element attributes would include line type, line width, and line color. An individual line attribute could be a LINE CAP, which specifies the appearance of the endpoints of a line element.

4.10.4.1.3 Input Primitives

Input primitives or logical devices are virtual devices designed to insulate the application from the real input devices. Logical devices include picking devices, locator devices, choice devices, valuator devices, etc. In terms of actual devices, a locator device might be associated with the first mouse button.

4.10.4.1.4 Input Model

The input model describes how input primitives and logical devices are related to physical input devices and the degree of control provided to the application over the devices. For example, one control choice might be how feedback is echoed to the operator when a logical locator device is attached to a depressed mouse button. The feedback might be a rectangular cursor or the highlighting of geometry as a cross-hair cursor moves over it. When the button is released, the device coordinates are placed in the locator data record, and an event is placed in an event queue for which the application can check asynchronously. The method the application uses to determine if a device has data for it is usually described in terms of modes. A common mode is event mode. The application waits a finite time for some event to appear in a queue. If no event comes in the finite time, the application does other processing and eventually comes back to check the queue with the wait for some event. If an event appears, the application determines what type it is and gets the data for that type of event. For a pick device, the data might be all possible graphical primitives that could intersect some aperture, possibly specified in the device coordinate system.

4.10.4.1.5 Coordinate Systems and Clipping

Part of the graphics services is a means to utilize various coordinate systems. Graphical output has to be described to the graphics system in terms of some coordinate system, relevant to the application and presented to the display device in terms of its own coordinate system, the device coordinate system. It is unlikely that these two coordinate systems will be the same. A graphics system may therefore involve a number of coordinate systems and hence the need to define transformations between them. Some standard types of transformations are scaling, rotating, translating, reflecting, and projection, such as parallel and perspective. They are used to manipulate objects in a coordinate system and to map from one coordinate system to another. The coordinate systems commonly used are modeling coordinates, world coordinates, view-reference coordinates, normalized projection coordinates, and device coordinates.

Clipping is the process of specifying a region in space and restricting graphical output to that region. Only those primitives that define objects in that region will have their output displayed.

4.10.4.1.6 Output Model

The output model is the concept of how graphics objects are created, displayed, and controlled on output devices. The output model defines how to position and organize objects on the screen and the visual state of these objects, such as visible or invisible, hidden lines removed or not removed, picture matches retained structure, picture not consistent with retained structure, etc.

More specifically, the output model concept is made up of the

- Transformation pipeline
- Rendering pipeline
- Retained structures
- Nonretained structures
- Graphics state
- Window systems

4.10.4.1.7 Storage/Archiving

Storage data formats for displayed or rendered images are required, but they are not treated by this version of this guide.

4.10.4.2 Graphics Services

The graphics services can be generalized as the ability to

- Create, delete, and modify output primitives
- Specify and edit the primitive attributes globally and individually
- Transform (i.e., scale, translate, rotate, reflect, project, etc.) primitives for construction of more complex objects and for arrangement in the viewing space
- Create and manipulate a database of primitives, to define and edit attributes, to create and combine transformations, and to control the display of graphics primitives selectively
- Display graphical objects constructed in a retained database, or the ability to display primitives immediately, or to display both from a retained database and immediately
- Apply lighting and shading algorithms to collections of graphical objects with multiple light types and sources
- Prepare display data and control the timing of the actual display of the display data; on some systems this is referred to as frame buffer control
- Store and retrieve graphical objects from files
- Control input devices and retrieve data from input devices
- Direct output to a metafile and retrieve graphics data from a metafile
- Enquire about all aspects of the graphics environment; e.g., the state of the system at any given time, the actual capabilities of a given hardware platform, the attributes and primitives supported by a given implementation, etc.
- Distribute graphics
- Control errors

4.10.4.3 API Services

The major categories of graphics services available in the POSIX OSE API area include

- Two-dimensional graphics API services

- Three-dimensional graphics API services
- Device interface API services
- Image processing API services

For most of these API standards there exist standard language bindings so that applications using different programming languages can access the same functionality.

The choice of which graphics standard API to use will depend on a number of factors: application profile, overall system architecture, equipment available, existing application database interaction, system performance considerations, user interface requirements, management policy, and other external factors. The aim of producing a compatible set of graphics standards in GKS, GKS-3D, PHIGS, PHIGS PLUS, etc. (described in 4.10.5.1) is to allow that choice to be made in the most flexible way.

4.10.4.4 EEI Services

The major categories of graphics services in the POSIX OSE EEI area include

- Protocols
- File formats

The choice of which standard to use depends on a number of factors: application profile, system architecture, equipment available, system performance considerations, and other factors.

4.10.5 Standards, Specifications, and Gaps

There are several major standards existing in the computer graphics industry today that have been approved by national or international organizations such as ANSI and ISO. There are also standards efforts going on in related areas such as application data exchange. These official graphics standards are complemented by de facto standards that have been accepted by the graphics industry at large. This subclause provides a general explanation of these standards, their specifications, and interrelationships.

See Table 4.21.

4.10.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

Table 4.21—POSIX OSE Graphics Services Reference Model Standards

Service	Type	Specification	Subclause
Two-dimensional graphics	S	ISO/IEC 9592 {56} (PHIGS), ISO 7942 : 1985 {20} (GKS), ISO/IEC 9636 : 1991 {61} (CGI)	4.10.5.1
Three-dimensional graphics	S	ISO 8805 : 1988 {37} (GKS-3D), ISO/IEC 9592 {56}, (PHIGS), ISO/IEC 9636 : 1991 {61} (CGI)	4.10.5.1
Image Processing	S	ISO/IEC 12087 {93} (IPI)	4.10.5.1
Protocols	P	PEX Specification {B49}	4.10.5.2.2
File formats	S	ISO/IEC 8632 : 1992 {29} (CGM), ISO/IEC 9592 {56} (PHIGS)	4.10.5.1
Conformance testing	S	ISO/IEC 10641 : 1993 {87}	4.10.5.1

NOTE — S, E, P, G-Standard, Emerging Standard, Public Specification, Gap

Table 4.22—Graphics Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
PHIGS		S			S		
GKS		S			S		
GKS-3D		S			S		
CGI		S					
IPI					E		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
PHIGS		S				
GKS	E	S		S		
GKS-3D	S	S		S		
CGI						
IPI						

NOTES:

- 1 — LIS—Language-independent specification is available.
- 2 — Ada, APL, Basic,... —Language-dependent specification is available.
- 3 — S, E, P, —Standard, Emerging Standard, Public Specification, Gap

4.10.5.1.1 Programmer's Hierarchical Interactive Graphics Standard (PHIGS)

ISO/IEC 9592 {56} defines the graphics standard; language-binding standards are contained in ISO/IEC 9593 {57}.

PHIGS is a functional specification of the interface between an application program and its graphics support system. It provides the following graphics functionality:

- A high degree of interactivity
- Multilevel, hierarchical structuring of graphics data
- Easy modification of graphics data and the relationships among the data
- Three-dimensional, as well as two-dimensional, graphical input and output
- Offline storage (and retrieval) of graphics data

PHIGS controls the definition, modification, and display of hierarchical graphics data, and it specifies functional descriptions of systems capabilities, including the definition of internal data structures, editing capabilities, display operations, and device control functions. PHIGS manages the organization and display of data in a centralized database, allowing programmers to define and organize graphical data in a manner most convenient to the application. Such a hierarchical approach is a benefit that is not available in GKS, another international standard.

Objects are defined in the PHIGS graphical database by a sequence of elements, including output primitives, attributes, transformations, and invocations of other object and object part definitions. These elements are grouped into entities called structures. Structures may be related in a number of ways, including geometrically, hierarchically, or according to inherent properties or characteristics, as defined by an application.

PHIGS provides tools to use hierarchical data structures with minimal effort by the application programmer. Pictures constructed from geometric models often have a clearly evident structure. This structure can sometimes be easily seen in the repeated use of symbols, in the connections and geometric relationships between objects, or in the overall organization of a complex image. Even if the structure of the object is not evident, its underlying data organization may be quite rigorous, well defined, and well understood by the application. PHIGS supports both these cases by separating the definition of graphics data from the actions required to display them.

The structured definition of graphics data inherently reduces repetition and connectivity problems. The repeated use of component objects and the relationships between them can automatically be made a part of the definition of an object.

The structured definition of data allows images to share component objects, making it faster and easier for application programs to define and modify picture descriptions. Sharing component objects will also reduce storage requirements for graphics data.

PHIGS permits rapid dynamic access to a centralized graphics database. This allows PHIGS to support interactive end-user application programs and, depending on the capability of the hardware, realtime definition and modification of graphics data. PHIGS is capable of performing three-dimensional modeling transformations, workstation transformations, and viewing. It also handles two dimensions through a shorthand functionality of three dimensions. In workstation transformations, PHIGS provides another level of display control after the viewing operation that can isolate a section of an image for pan and zoom operations.

NIST has developed a test system to help determine whether implementations of PHIGS conform to the specifications of ANSI X3.144.²⁰ The PHIGS Validation Test (PVT) suite consists of highly portable Fortran programs that examine test conditions and report the results.

ISO/IEC 9592-4 :1992 specifies a set of extensions that addresses some of the deficiencies in the graphics functionality provided by ISO/IEC 9592-1 :1989, ISO/IEC 9592-2 :1989, and ISO/IEC 9592-3 :1989. PHIGS does not include “higher level” primitives, such as curves and surfaces, and techniques for lighting and shading. Recognizing this, an ad hoc working group was formed to propose a set of extensions to PHIGS to enable these capabilities to be addressed in a standard manner, compatible with the overall philosophy of PHIGS. This set of proposed extensions was submitted to ISO and has since been developed into PHIGS PLUS. PHIGS PLUS enhances PHIGS by providing

- Primitives for defining curves and surfaces
- Lighting models
- Shading of surfaces
- Depth cueing
- Color mapping and direct color specification

4.10.5.1.2 Graphical Kernel System (GKS)

ISO 7942 :1985 {20} (also FIPS Publication 120-1 {B16}) defines the GKS; language-binding standards are contained in ISO 8651 {32}.

GKS is a two-dimensional graphics system and provides no support for three-dimensional graphics. It is an API that shields the programmer from differences among various computers and graphic devices. It allows for portability of graphics applications by standardizing the basic graphic functions and the method and syntax for accessing these functions.

GKS is widely used today. It has standard language bindings for Fortran, Pascal, C, and Ada. A language binding for Lisp is currently in development.

²⁰ Please note that this standard has been withdrawn and replaced by ISO/IEC 9592 {56}.

GKS supports the grouping of logically related primitives such as lines, polygons, strings, and their attributes into collections called segments, which cannot be nested.

GKS supports many graphical input and output devices such as black/white and color displays, printers, plotters, mice, data tablets, joysticks, and digitizers.

4.10.5.1.3 Graphical Kernel System for Three Dimensions (GKS-3D)

ISO 8805 : 1988 {37} defines GKS-3D; language-binding standards are contained in ISO/IEC 8806 {38}.

GKS-3D specifies extensions to GKS for defining and viewing three-dimensional wire-frame objects. In addition, the GKS input model has been extended to provide three-dimensional locator and stroke input. GKS-3D allows the operator to obtain information from three-dimensional input devices and to perform hidden line/hidden surface removal (HLHSR) at the workstation. It does not, however, provide specific functions for controlling rendering techniques such as light source, shading, texturing, and shadow computations that have to be done at the workstation. Conceptually, all workstations are three-dimensional in GKS-3D, which is made possible by shielding the hardware peculiarities as in GKS.

4.10.5.1.4 Computer Graphics Interface (CGI)

ISO/IEC 9636 :1991 {61} defines CGI; language-binding standards are contained in ISO/IEC 9638 :1994 {63}; encoding standards are contained in ISO/IEC 9637 {62}.

CGI specifies a standard functional and syntactical specification of the control and data exchange between device-independent graphics software and one or more device-dependent graphics device drivers. Unlike the graphics standards discussed earlier, CGI specifies an interface at the device-driver level, rather than at the application level.

Unlike CGM, which only handles graphical output, CGI handles both input and output, which makes all devices appear as identical graphics devices. It provides a standard graphics escape mechanism to access nonstandard graphics device capabilities. CGI allows programmers to write portable device-driver software that is independent of the physical graphics device characteristics. This makes the software portable and compatible with a wide variety of devices.

4.10.5.1.5 PHIGS Archive Files

Parts 2 and 3 of the PHIGS standard (ISO/IEC 9592 {56}) define an archive file format for storage and transfer of PHIGS structures and structure network definitions from the CSS (Central Structure Store). Part 2 describes the file format and Part 3 a clear text encoding. This encoding is constructed using the same techniques as used by CGM.

4.10.5.1.6 Image Processing and Interchange (IPI)

ISO/IEC 12087 {93} defines IPI; language binding standards are contained in ISO/IEC 12088 {94}.

IPI is a functional specification and several language bindings for an API to imaging. The standard defines the data objects, primitive operations, and a reference model. The API supplies the basic building blocks upon which applications requiring imaging functionality can be built within conventional, distributed, and image-oriented computing environments.

IPI includes three parts:

- Part 1 Common Architecture for Imaging (CAI)
- Part 2 Programmer's Imaging Kernel (PIK)
- Part 3 Imaging Interchange Facility (IIF)

4.10.5.2 Additional Specifications

4.10.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

4.10.5.2.1.1 Conformance Testing of Implementations of Graphics Standards

ISO/IEC 10641 :1993 {87} specifies an approach for testing the conformance to computer graphics standards of products that claim to implement these standards. It addresses the conformance testing processes for all classes of graphics standards.

The main reasons for introducing a standard on conformance testing in the area of computer graphics are

- To promote standards that are developed in a way such that products can be tested for conformance to the requirements of that standard
- To promote addressing conformance in standards
- To promote high-quality test suites that are appropriately defined to test products for conformance to all areas of the standard
- To promote test methods that are developed in a consistent way for similar standards
- To promote conformance testing that is carried out in a consistent way throughout the international graphics community

4.10.5.2.2 Public Specifications

The specifications listed in this subclause are not part of the POSIX OSE, but may be of interest. Use of these specifications should be considered carefully because there are risks in using these unapproved specifications. They are included in this guide to indicate some of the existing work that has been done in areas that are gaps in the POSIX OSE.

4.10.5.2.2.1 PHIGS Extensions to X (PEX)

PEX {B49} is a network protocol extension to the X Window System. Because many applications require three-dimensional graphics, which is not supported by X, it became necessary to extend the X Protocol to include three-dimensional graphics. PHIGS was selected as the API because of its acceptance as a three-dimensional standard, its high degree of input ability, and its powerful database editing capabilities. In 1988, the MIT X Consortium contracted to add three-dimensional and extended input extensions to the X Protocol. The first release of PEX as a sample implementation (PEX-SI) was made in January 1991, and it is now available commercially.

4.10.5.3 Unaddressed Services

Current standards do not address the following services:

- Current standards allow a wide interpretation for implementors of the standards, thus denying the applications useful controls. In order to achieve true portability in a distributed environment, applications will need control and deterministic functionality.
- Current standards do not address solid modeling.
- Current standards do not allow drag-and-drop support with dynamic linking of live data between applications.
- Current standards do not allow nonretained graphic methods to do lighting and shading.

4.10.6 POSIX OSE Cross-Category Services

Not applicable.

4.10.7 Related Standards

4.10.7.1 IGES

ANSI/ASME Y14.26M-1989 {112} defines IGES. See 4.5.

4.10.7.2 IEEE Std 1295-1993 {127}

IEEE Std 1295-1993 {127} defines the X Window System graphical user interface Modular Toolkit Environment (MTE). See 4.9.

4.10.7.3 SGML

ISO 8879 : 1986 {44} defines SGML. See 4.5.

4.10.7.4 IGES/PDES Organization (IPO)

See 4.5.

4.10.7.5 ISO/IEC TC184/SC4 (STEP)

See 4.5.

4.10.7.6 CGM

ISO/IEC 8632 : 1992 {29} defines CGM. See 4.5.

4.10.8 Open Issues

The following are open issues with respect to graphics services.

- Applications have different behaviors for similar functions, which hinders user portability. Standards are needed for a uniform approach (for example, a graphics style guide) so that users can switch between applications without a lot of training.
- The relationship between window standards and CGRM is unclear.
- There are no standardized ways to cut and paste between applications.

4.11 Application Software Development Support Services

4.11.1 Overview and Rationale

The services described in this clause enable application software to be physically developed and executed on the application platform.

Most computers come with the capability to develop applications. These tools traditionally are text editors, compilers, and linkers.

4.11.2 Scope

The scope of this clause is to address the requirements and associated standards for application development using the development tools available on a typical computer. These are the traditional tools that have been used on most computer systems for many years. This is an important area of information systems and would benefit from standardization efforts.

An area of application development that is not currently described in this clause is that of Software Development Environments (SDE). SDEs often have sophisticated capabilities to specify, plan, track, develop, test, and maintain extremely large and complicated software projects throughout the complete software life cycle. For more details on SDE standardization activities, see 4.11.7.

4.11.3 Reference Model

This subclause identifies the interfaces for application software development support services that impact application portability or system interoperability. Figure 4.15 shows these interfaces.

This reference model is consistent with and expands on the reference model in Section 3.

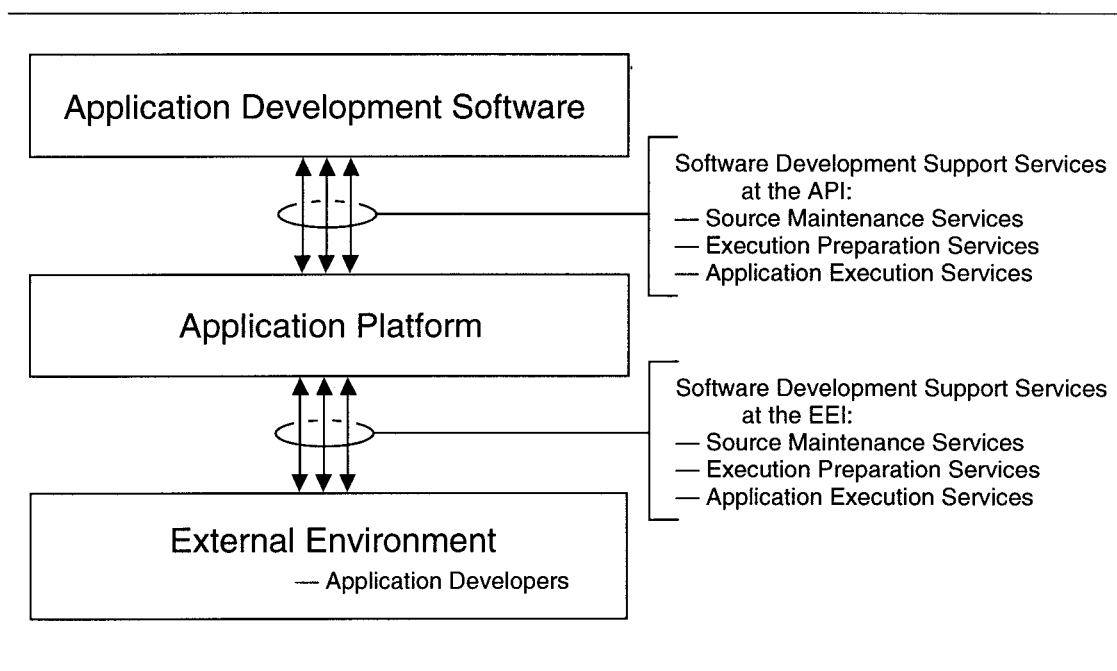


Figure 4.15—POSIX OSE Application Software Development Services Reference Model

4.11.4 Services

4.11.4.1 API Services

This interface will allow an application to access all of the services available to the user at the EEI.

4.11.4.2 EEI Services

These services are those that allow traditional applications to be developed. Many tools are available on existing computer systems that perform these services.

4.11.4.2.1 Source Maintenance Services

These services allow an application developer to create and maintain applications. They include the ability to

- Prepare the source of an application with a syntax-directed tool. This tool should have features such as the ability to
 - Find references to variables, routines, data structures, etc.
 - Check for syntax errors and assist the user in fixing those errors
 - Attach commentary to the source that is not part of the source
- Track revisions to software effectively, including the ability to
 - Check software in and out for modification
 - Determine the status of software under development
 - Query the modification history of software
 - Create complete revision history of all changes to software
- Support development styles, including the ability to
 - Optionally enforce development style rules
 - Reformat source to conform to a given set of style rules
 - Handle multiple sets of stylistic rules
- Examine source automatically and display potential logic errors

4.11.4.2.2 Execution Preparation Services

These services allow an application developer to prepare an application for execution. They include the ability to

- Prepare the source of an application for execution on the application platform. This includes the ability to specify execution options where necessary. (On many systems this is the traditional compilation and linking capability.)
- Generate programs from multiple source modules automatically
 - Determine automatically which source has been changed
 - Determine automatically the dependencies between source modules
 - Generate previous versions of the software

4.11.4.2.3 Application Execution Services

These services can be used by an application developer to work with running applications. They include the ability to

- Debug an application, including the ability to
 - Run an application under complete control of a debugger
 - Attach the debugging tool to an application while it is executing or after it has failed
 - Produce a cross-reference listing of variables, routines, data structures, etc.
- Examine program execution efficiency (e.g., profiling)
 - By routine or major block of source
 - By individual lines of source

4.11.5 Standards, Specifications, and Gaps

See Table 4.23.

Table 4.23—Application Software Development Support Standards

Service	Type	Specification	Subclause
Source code maintenance	S	ISO/IEC 9945-2 :1993 {69}	4.11.5.1
Execution preparation	S	ISO/IEC 9945-2 :1993 {69}	4.11.5.1
Application execution	G	n/a	4.11.5.3

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 4.23 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE. (The entry for ISO/IEC9945-2 : 1993 {69} applies only to the API portion of that standard.)

Table 4.24—Application Software Development Support Services Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
ISO/IEC 9945-2 :1993 {69}	E				S		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
ISO/IEC 9945-2 :1993 {69}						

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

4.11.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

4.11.5.1.1 ISO/IEC 9945-2: 1993 (IEEE Std 1003.2-1992) {69}

This standard describes limited tools that support application development, including standard interfaces to selected compilers and software development utilities.

4.11.5.2 Additional Specifications

None.

4.11.5.3 Unaddressed Services

While some of the services are addressed in the standards cited in this clause, application development services are not thoroughly covered in any of the standards. Examples of unaddressed services include

- Syntax-directed editing
- Coding style support
- Configuration management
- Source maintenance
- Application execution services
- Interoperability of software development environments

X/Open XPG4 {B69} and OSF AES-OSC {B50} provide specifications for some of these unaddressed services.

4.11.6 POSIX OSE Cross-Category Services

None.

4.11.7 Related Standards

While SDEs are not currently described in this clause, there are standardization efforts underway in this area. One forum that is addressing the issue of SDEs is the NIST Integrated Software Engineering Environments (ISEE) effort. Another is the Portable Common Tool Environment (PCTE), ISO/IEC 13719-1 :1995 {B4} .

5. POSIX OSE Cross-Category Services

The POSIX OSE reference model defines a set of conceptual system building blocks that collectively describes the POSIX OSE. Each building block provides a specific set of interfaces for access to their associated facilities and services. There is another class of services and requirements, however, that may influence or impact, or both, the basic architectural building blocks; these are referred to as POSIX OSE cross-category services.

A POSIX OSE cross-category service is a set of tools and/or features that, when applied, may have a direct effect on the operation of one or more of the open system components, but it is not in and of itself a stand-alone OSE component. Examples of POSIX OSE cross-category services include internationalization, security and privacy, administration, etc. Internationalization has a number of attributes that influence multiple OSE components; supporting multiple coded character sets, for example, will affect end-user interfaces, operational message input and output, screen display, data collating sequences in programming languages and database systems, etc.

This section deals with the general characteristics of POSIX OSE cross-category services as applied to the POSIX OSE architectural components and to the profiles and domains that characterize application environments. The specific impact/influence of a POSIX OSE cross-category service is described in the appropriate subclause of Section 4 that deals with individual OSE components.

This section describes issues that should be considered in writing profiles, and is organized so that subclauses for each POSIX OSE cross-category service points to, and addresses issues adjacent to, each of the service categories identified in Section 4. Section 5 is issue oriented, while Section 6 addresses the profiling process. These issues define areas that need to be merged to arrive at a balanced solution for a specific profile. It is expected that the specific merges would be made by the profiler, but that this section could give guidance for merging and could also be used to accumulate lessons learned.

5.1 Internationalization Services

5.1.1 Overview and Rationale

Historically, information systems intended for use within a particular national or cultural market have been designed specifically for the requirements of that market. If the vendor or developer was based in a country other than that of the target market, this was typically accomplished through substantial re-engineering of the features of an existing system designed for some other country, and doing so at considerable cost. As the developer desired to market the system in additional countries, the process of re-engineering was repeated for each new national or cultural market. Application software developers were faced with the same problem. The very nature of this style of development produced little concern for portability across national or cultural boundaries or for interoperability between them. Users or organizations that needed to operate in multiple national or cultural markets typically did so with multiple, generally incompatible, information processing systems.

The interfaces provided by the POSIX OSE can be generalized or abstracted through the use of internationalization, however, to extend across national and cultural boundaries. Such a model provides the foundation for international portability of application software, increased user portability, and enhanced interoperability and data exchange capabilities. The task of internationalization is to ensure that the services provided by the POSIX OSE, and the interfaces between such services, are specified in such a way that they can be easily used all over the world. Additionally, as the user is likely to require services from any or all of the service categories of the POSIX OSE, internationalization impacts all areas of the POSIX OSE and should be viewed as an OSE cross-category service. Since the internationalization aspects of general OSE services and API services are similar for all of the OSE service categories, they are discussed here rather than repeating them in each of the services sections within this guide.

The ability of the service categories of the POSIX OSE to support multiple natural languages, and the underlying cultural conventions, is a two-step process. These two steps are generally referred to as internationalization and localization. First, the interfaces between the service categories are generalized so that they are not oriented to the requirements of any particular natural language or set of cultural conventions (internationalization). Then, facilities are provided by the POSIX OSE that allow the user to select the desired natural language and cultural conventions (localization). Services are provided to facilitate this process.

Within this context, cultural conventions, while discussed more fully later in this clause, may be viewed as various aspects of how information is presented to the user. Different cultures, for example, use different formats for dates and numeric values and use different currency symbols. The interfaces provided by the POSIX OSE should allow the information to be presented to the user in the appropriate format as well as the appropriate natural language.

5.1.2 Scope

The POSIX OSE addresses services that are necessary to support users, irrespective of their particular natural language or cultural conventions. While it is not expected that every implementation of the POSIX OSE would provide support for all possible natural languages and cultural conventions, the specification of the services and the interfaces within the POSIX OSE should not preclude such support. In addition to the service and interface requirements described here, it should be noted that internationalization is affected by a number of elements that are beyond the scope of this guide. Actual implementations of the internationalized POSIX OSE, for example, may need to consider the impact of multiple sets of governmental and regulatory agencies, international data communication standards, and other elements that are presently not specified within the POSIX OSE, such as data portability between localized information processing systems.

Services differ from country to country and even between users within one country. Many users, for example, may require the simultaneous support of multiple natural languages and cultural convention sets. Therefore, the basic internationalization requirement within the POSIX OSE is to provide a set of services and interfaces that allow the user to define, select, and change between different culturally related application operating environments supported by the particular implementation. Specifically:

- The POSIX OSE should provide the means of adjusting the output of specific functions and utilities to support different natural languages, cultural conventions, and character sets as may be required by the supported natural languages.
- A user should have the capability to select an internationalized user environment that specifies a particular set of data presentation characteristics, including cultural conventions, character sets, and native language.
- It should be possible for implementations of profiles of the POSIX OSE to support different applications functioning in different internationalized user environments concurrently, supplying different sets of natural languages, cultural conventions, and character sets for different users.
- The capability of supporting different internationalized user environments, and their associated natural languages, cultural conventions, and character sets, should not require any changes to the logic of properly architected application programs.
- The effect of the user selecting a new internationalized user environment, and its associated natural language, cultural conventions, and character set, should be transparent to application programs.
- The model should be flexible, to support future extensions and requirements.

5.1.3 Reference Model

Internationalization is a POSIX OSE cross-category service, spanning all POSIX OSE service categories. While various reference models have been used in published technical papers to depict internationalization issues, the internationalization services described in this clause conform to the OSE reference model.

5.1.4 Services

The POSIX OSE should provide services on different levels: general services to be satisfied for any requesting program; API services to be satisfied at the API for a specific program; and a set of tools to support the localization of systems and applications. This subclause will discuss these different services in detail. In examining these services, it is helpful to draw a distinction between those services that are required to support the portability of an application platform across cultural boundaries and those services that are required to support the portability of an application across one or more sets of cultural conventions that may be supported on a single application platform.

5.1.4.1 General Services, Application Platform

Internationalization services are focused on support and handling of

- Character sets and data representation
- Cultural conventions
- Natural language support

5.1.4.1.1 Character Sets and Data Representation

The character set for the English language can be satisfied by the ISO 646 :1991 {1} character set, which uses 7 b to identify uniquely each of the 95 available characters. For European and American languages besides English, the number of local characters is much larger. The Far Eastern language requirements for thousands of pictograms add yet another dimension to the coding rules and techniques.

Different standards address the methods by which the local character repertoires can be coded for unique identification. While replacement of seldom-used characters in the 7 b codings can support a single additional language besides English, 8 b coding schemes are used to satisfy multiple languages concurrently by assigning an additional 96 graphic characters to the available repertoire. An example is ISO 8859-1 :1987 {43}, which can support languages of Western Europe, America, Australia, and other English speaking countries. For Eastern Europe, Greece, Russia, Arabia, and many other countries, other 8 b codes are defined. Japan, China, Korea, and Taiwan have so many characters in their repertoire that 16 b or more are needed to identify them clearly.

Because different coding schemes are used, it is important that the application platform have the potential capability of supporting all of them. It is also important that the application platform have the capability to represent (display, print) the data correctly. It is also important that an application be able to determine in which coded character set data items are stored on disk or tape. Otherwise, it is impossible for the application to interpret the data correctly. Currently the user needs to control the consistent use of the same coded character set within an application, but in the future the application platform should be able to provide identification methods for the coded character sets used for data storage, processing, communication, and presentation. It might also be advantageous for the application to be able to prohibit users from updating data stored in one coded character set with data in another coded character set, since this would immediately corrupt data bases or fiat files. Therefore, it may be necessary in the future to provide a method of announcing the coded character set in which data are stored, processed, communicated, and presented. This includes single-, double-, or quadruple-octet character code sets.

The general service for support of character sets and data representation in an international environment are

Coded character set independence

The application platform should be able to input, store, manipulate, retrieve, communicate, and present data independent from the coding scheme used. This includes 7 b, 8 b, 16 b, state-dependent, and multi-octet coded character sets.

Character set repository

The application platform should be able to maintain and access a central character set repository. This repository contains all coded character sets used throughout the platform and specifies relevant information about them.

Code format

The application platform should be able to contain the format of the character code: 7 b, 8 b, 16 b, or any other format.

Data class definition

An application needs to be able to inquire about a variety of properties of the characters that may vary from locale to locale. Examples of properties include alphabetic or numeric.

Collating rules

Since different character sets have different coding for characters, the application platform should be able to compare strings of such coded characters following rules defined for the specific character set. Culturally dependent additional collating rules are discussed in 5.1.4.1.2.

Lower- to uppercase mapping

A mechanism should be provided to perform extensible character mappings for satisfying various local language needs. The application platform should be able to handle rules of character mapping such as lowercase-to-uppercase and uppercase-to-lowercase. For some specific characters, no upper- or lowercase is available. Examples are the lowercase umlauts, which do not have uppercase representations in Switzerland; the uppercase forms are A, O, or U, respectively, followed by a lowercase “e.” This concept need not exist in some cultures. Some cultures have other rules of character mapping than uppercase or lowercase mapping. For example, Japanese kana has a mapping rule between Hiragana and Katakana.

Character directionality rules

Some languages, like Hebrew and Arabic, are written from right to left; numbers within text in these languages are written from left to right. The application platform should be able to store these directionality rules with the character set.

Presentation rules

The application platform should be able to provide fallback presentation rules for the presentation of coded characters that have no associated graphic shape.

Character set identifier

The application platform should be able to identify uniquely each coded character set to allow compatibility checks and translation or transliteration to and from other registered character sets. This ensures data integrity in the communication of data across computers and networks.

Character set selection

The application platform should allow the end user or the application to select the coded character set to be used; otherwise, the application should automatically select a default coded character set according to preset parameters. It should be possible to switch to other coded character sets and to invoke translation routines where required.

Data announcement

The application platform could benefit from having the ability to recognize the coded character set of data entities (files, messages, etc.). One way of doing this is to store the character set identifier together with the data; standardization efforts are under way to formalize this process, with consideration being given to the level of granularity of such identification (e.g. file, word, character). The announcement enables the application to prohibit updates with data coded in other character sets, thus ensuring data integrity even in distributed systems.

Data presentation

The application platform should be able to present data on different display or output devices, potentially according to rules in a repository, including escapement of characters and selection of

different shapes. Preparing data for presentation may involve extensive translation and transliteration due to potential hardware limitations of the printers and displays used in a particular installation.

Data communication

The application platform should be able to transmit and receive data from communication systems and to maintain the integrity of the information. In an internationalized environment, this capability might include data translation due to different coded character sets being used by different service categories of the application platform.

Data input

The ability to enter data is not necessarily controlled by the application platform. In the future, it might be necessary to provide a mechanism to input multiple local character sets in the same text. It might then be beneficial for input data to carry some form of character set identification.

5.1.4.1.2 Cultural Conventions

Besides using different characters and different languages, countries throughout the world have also developed quite different cultural conventions. Even within one country we can find significantly different cultural environments. The prime example is Switzerland, where French, German, Italian, and Rhaeto-Romanic are officially accepted languages. Combined with the language preferences are conventions about the formats of time, date, numeric values, and measuring systems. Currency symbols, paper formats, hyphenation, and collating are dependent on cultural conventions. End-user-oriented applications have to address these issues to provide a familiar local view, which helps to prevent operating errors.

The general services for cultural conventions are

Cultural convention repository

The application platform should have the ability to store and access rules and conventions for cultural entities. These might be areas with a common language, geographic areas, or areas with common cultural or historic background. The repository should contain specifications and presentation rules for

Date and time formats

These indicate the formats associated with the particular cultural entity. For example, while in the US the date is expressed in the format month/day/year, the European preferred format is year-month-day for data processing purposes and day-month-year in personal use. Japan counts the years according to the reign of the current emperor. Additionally, 24 h clocks, which are prevalent in Europe, are commonly used only in military circles in the US, while the terms “am” and “pm,” denoting morning and afternoon, are used by the general public. These are only a few examples for the cultural differences in this area. The application platform should be able to store the preferred forms for date and time for a specific cultural entity and make it available upon request in this format.

Week and day numbering

In Europe, the week starts on Monday; in the US, on Sunday. The application platform should be able to supply the requesting program with the needed information, potentially from a repository according to specified rules.

Formats of numeric fields

The handling of numeric fields in unfamiliar formats is one of the major reasons for human errors. The application platform should provide the service to format the values according to specifications in the repository. The characters that signify the decimal point (comma, period, etc.) should be defined, as well as the number of decimals, the grouping of digits before the decimal point and the presentation of negative values.

Currency symbols and field length

The handling of currency symbols in the different cultural areas should be provided by the general internationalization services. The currency symbols might be more than one digit long and can appear before or after the currency field. The format of currency fields might differ from that of numeric fields; for example, in Portugal the \$-sign is used as the decimal point. Information about these conventions should be stored in the repository and be used by the

application platform for local formatting of currency fields. Not necessarily a service, but similarly important, is the understanding that, due to the value of different currencies, the field lengths should be considered carefully. Also some currencies do not have decimals (e.g., Italian lira).

Paper formats

Internationally usable and portable applications should be able to print on different paper formats. While quarto format is predominant in the US, the ISO standardized A-formats are used in Europe, Korea, Hong Kong, and Japan. Printer drivers should be able to adjust their output to local formats, defined in the cultural convention repository.

Cultural repository selection

These repositories should be available to all applications. Users and applications should be able to select a repository from the application platform; a default value should be provided if no selection is made. An additional service allows dynamic switching to other repositories upon user or program request.

Collating rules

Besides the generic binary and character-set-dependent sorting rules, the application platform should have the ability to sort data according to local rules, defined in the repository. An example for culture-dependent collating rules is the handling of umlauts; while they are sorted with the base characters in Austria, they are sorted at the end of the alphabet in Sweden. Adding complexity, they can be sorted differently within one country between normal business use, such as dictionaries, and in telephone books. Other idiosyncrasies are the sorting of one character as two (the German “sharp-s” sorts as “sz” in Austria and “ss” in Germany), or two characters as one, or the position of accented characters in a string, and more. User-defined collating tables in the cultural convention repository allow culture- or application-dependent sorting services.

5.1.4.1.3 Natural Language Support

The POSIX OSE should give users the ability to select a natural language for their dialogue with the system and applications. While it is unrealistic to expect all application platforms to support all possible natural languages, error messages, online documentation and help facilities, selection menus, and the relevant user interaction with these services should be prepared for translation into the supported user-selectable natural language. Additionally, the POSIX OSE should support differences between the natural language selected by the user for interaction with the application platform and that selected for use within a particular application. For word- and text-processing, the service includes hyphenation and spell checking with possible thesaurus support in different languages. The problem is complicated by the fact that data can contain text in different languages in the same document.

The services for natural language support are

Multiple language capability

The application platform should be able to support more than one language simultaneously. For example, one process might be providing French language capabilities while another process operated in Japanese. The application platform should be able to let users select their preferred languages for communication with the application and allow them to switch dynamically to another language. The application platform also should have the capability to assign a default language, based on parameters for the application platform, the specific workstation, the user identification, or the application.

Natural-language message system

The application platform should have the capability to present (display, print, etc.) messages, menus, forms, and online documentation in the language selected by the user. The application platform should be able to support multiple languages simultaneously for different users, and it should allow the user to switch from one language to another. The following problems also should be handled correctly:

Natural-language independence

The elements of program code of an application should be able to be independent from any particular natural language, presenting messages in the natural language used within the internationalized user environment selected by the user.

Variable message length

The application platform should support the presentation of messages of variable length, as translation into other languages changes the length of the message; English text is usually quite short compared to the same text in, e.g., German or Finnish. Ample room should be available in the display field to accommodate this variation.

Inserted parameters and word order

The application platform should be able to insert variable parameters into messages at the location appropriate for the user-selected natural language.

Support of local keyboards

The application platform should be able to interpret correctly the input from keyboards that have been modified locally to support the local character sets.

Local language user interaction

The application should be able to accept solicited input from the user in the language selected by the user, without dependence within the application logic on a particular natural language or set of cultural conventions. For example, many applications use the first characters of prompts to make selections; this method is not acceptable in an internationalized system. The translation process changes the prompts and with them their first character; more than one prompt could have the same start-character and the program logic would not work. Multiple languages should be supported simultaneously.

5.1.4.2 API Services

All the general services defined in 5.1.4.1 should be accessible from the applications through requests to the API. The API services can be structured in the same way as the general services for which they call.

5.1.4.2.1 Cultural Conventions

Cultural convention invocation

The application platform should allow the application to invoke a specific cultural convention from the repository. It should automatically invoke the default convention set, if no selection is made by the application.

Cultural convention change

When requested by the application or the user, the application platform should change the used cultural convention dynamically.

Provide local values

When requested by the application, the application platform should return local formats for time, date, calendar, numeric fields, currency fields, and symbols.

Local sort and comparison

When requested by the application, the application platform should compare and sort data according to the local collating rules defined in the cultural convention repository.

5.1.4.2.2 Natural Language Support

Language selection

The application platform should present messages, menus, forms, online documentation, and user interaction in the natural language selected by the user or automatically by the system, based on preset parameters for the application, the session, the user, or the system.

Change of language

Upon request from the user, the application platform should be able to change dynamically, prior to the invocation of a particular user application, the language used for messages, menus, forms, online documentation, and user interactions.

5.1.4.3 Localization Tools

Internationalization of application platforms and applications is the basis for their localization in the different countries. It is important for the user that this localization can be performed in a well prepared, organized way without the need to know the internal structure of the application platform or the application. The following services for localization tools are key to successful localization of application platforms and applications:

Character set repository services

Services should be provided to set up and maintain character set repositories. They also should allow the addition of new character sets to the repository.

Cultural convention repository services

Services should be provided to set up and maintain the cultural convention repositories. Addition of new cultural environments should be possible. User-definable collation tables are essential parts of these repositories; services to define and maintain them should be offered.

Translation support services

Services should be provided to set up and maintain local language message files, menus, forms, online documentation, and user interaction tables. The addition of new supported languages should be allowed by such services. Additionally, any such translation service should allow revision control, so that only new or changed text would require translation for new software releases.

5.1.5 Standards, Specifications, and Gaps

There are not many standards available that deal with internationalization. The majority of current standards describe character sets, both for control characters and for graphic characters in different coding schemes (7 b, 8 b, etc.). A few standards address the formats of time and date, and some standards touch peripherally on the subject of data announcement.

An example of how cultural conventions and languages are currently supported is the *setlocale()* function. It allows the application developer to select portions or all of predefined support features for national languages and local cultural conventions. The portions, called categories, correspond to the areas of functionality; presently supported are character classification, collation sequence, date/time format, monetary format, and numeric format. Other categories, such as message handling, are also likely to be implemented.

See Table 5.1.

Table 5.1—Internationalization Standards

Service	Type	Specification	Subclause
Character set/ data representation	S	ISO 646 :1991 {1}, ISO/IEC 2022 :1994 {5}, ISO 4031 :1987 {7}, ISO 4217 :1990 {8}, ISO 4873 :1991 {9}, ISO 6093 :1985 {10}, ISO/IEC 6429 :1992 {12}, ISO 6936 :1988 {14}, ISO/IEC 6937 :1994 {15}, ISO/IEC 7350 :1991 {17}, ISO 8601 : 988 {26}, ISO 8859 {43}, ISO/IEC 10367 :1991 {84}, ISO/IEC 10646-1 : 1993 {88}, ITU-T T.61 {100}, GB 2312-1980 {115}, JIS X0208-1990 {132}, KS C 5601-1987 {134}	5.1.5.1
Cultural convention	S	ISO 2014 : 1976 {4}, ISO 3307 : 1975 {6}	5.1.5.1
Natural language support	S	ISO/IEC 9995 : 1994 {70}, CAN/CSA-Z243.200-92 {B8}	5.1.5.1
NOTE — S, E, P, G—Standard, Emerging Standard, Public Specification, Gap			

5.1.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

5.1.5.1.1 International Standards

ISO 646 : 1991 {1}

Defines the binary representation of 128 control, (Latin) alphabet, digit, and symbol characters. Describes in general the use of the control characters. Describes option of national replacement characters.

ISO 2014 : 1976 {4}

Specifies the writing of dates of the Gregorian calendar in all-numeric form, signified by the elements year, month, and day.

ISO/IEC 2022 : 1994 {5}

Defines techniques for expanding the number of characters represented by the base character set.

ISO 3307 : 1975 {6}

Establishes uniform time representation based upon the 24 h timekeeping system. Provides a means for representing local time of the day and Universal Time in digital form for the purpose of interchanging information among data systems.

ISO 4031 : 1987 {7}

Specifies a standard means for representing local time differentials to facilitate interchange of data among data systems.

ISO 4217 : 1990 {8}

Specifies the representation of currencies and currency symbols.

ISO 4873 : 1991 {9}

Outlines the structure of the ISO8 b code and rules for implementation.

ISO 6093 : 1985 {10}

Specifies three presentations of numerical values, which are presented in character strings in a form readable by machine, for use in interchange between data processing systems. Also provides guidance for developers of programming language standards and implementors of programming products. These representations are recognizable by humans, and thus they may be useful in communication between humans.

ISO/IEC 6429 : 1992 {12}

Defines control functions and their coded representations for use in a 7 b code, an extended 7 b code, an 8 b code, or an extended 8 b code. Specifies a CO set, a C1 set, control functions derived from these sets, and a number of independent control functions.

ISO 6936 : 1988 {14}

Specifies the rules for conversion between the ITA 2 representation of 58 characters and the ISO 646 : 1991 {1} representation of 128 characters.

ISO/IEC 6937 : 1994 {15}

Defines terms and concepts used in describing and using code representations of character sets. Defines a repertoire of Latin alphabetic and nonalphabetic characters. Specifies binary representation of the characters. Specifies rules for the definition and use of character sets that are subsets of the repertoire.

ISO/IEC 7350 : 1991 {17}

Specifies the procedures for preparing, registering, publishing, and maintaining the register of graphic character sets that are composed from the character repertoire of ISO 6937 : 1994 {15} and the procedures for assigning identifiers to the sets.

ISO 8601 : 1988 {26}

Specifies the representation of A.D. dates in the Gregorian calendar and times and representation of periods of times. Applicable whenever dates and times are included in information interchange.

ISO 8859 {43} Specifies a set of up to 191 graphic characters by means of a single 8 b byte. The parts (“-x”) indicate different coded character sets:

-1 Latin Alphabet No. 1

-2 Latin Alphabet No. 2

- 3 Latin Alphabet No. 3
- 4 Latin Alphabet No. 4
- 5 Latin/Cyrillic Alphabet
- 6 Latin/Arabic Alphabet
- 7 Latin Greek Alphabet
- 8 Latin/Hebrew Alphabet
- 9 Latin Alphabet No. 5

ISO/IEC 9995 : 1994 {70}

This family of standards defines the layout of keyboards so that they can be used for input of multilingual information.

ISO/IEC 10367 : 1991 {84}

Specifies a unique graphic character set for use as GO set and a series of coded graphic character sets of up to 96 characters for use as the G1, G2, and G3 sets in versions of ISO 4873 : 1991. All sets specified in this standard are shown as elements of an 8 b code.

ISO/IEC 10646-1 : 1993 {88}

Defines the presentation of all of the scripts of the world in computer-based systems and their unambiguous interchange between one system or person and another. It is applicable to the representation, processing, storage, and presentation of the written form of the languages of the world.

ITU-T T.61 {88} Describes detailed definitions of the repertoires of graphic characters and control functions to be used in the international Teletex service. The means by which supplementary character repertoires are defined are also described.

5.1.5.1.2 Regional Standards

Presently, no regional internationalization standards that relate to the scope of this guide have been adopted.

5.1.5.1.3 National Standards

Many of the international standards have corresponding national standards; i.e., the same text is given a local standard identification. Also, national standards bodies have often developed standards for local representation of time, date, and currency. The implementation of these standards into an internationalized system is a prime example of localization.

Some of the standards that have no international equivalents are

GB 2312-1980 {115}	Chinese national character set standard
JIS X0208-1990 {132}	Japanese national character set standard
KS C 5601-1987 {134}	Korean national character set standard

5.1.5.2 Additional Specifications

5.1.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered cautiously as there may be risks in using these emerging standards prior to their final approval.

5.1.5.2.1.1 International Standards

The rapid development of business opportunities in the Pan-European and the Asian market has spawned a wealth of activities to develop standards for the support of internationalization in the field of information technology. These emerging standards deal with character sets, language neutral user interfaces, and communication.

IEEE P1003.2b {B24}

The IEEE P1003.2 working group is defining extensions to ISO/IEC 9945-2 : 1993 {69}. They include extensions to internationalization features and additional code conversion utilities.

PDAM to ISO/IEC 9899 {67}

This amendment defines the extensions to the C language standard to handle multibyte characters.

5.1.5.2.1.2 Regional Standards

The European Community is in the process of defining European standards, called EN (Européen Norm). No internationalization standards have yet been adopted.

5.1.5.2.1.3 National Standards

National standards under development that relate to internationalization include

CAN/CSA-Z243.200-92 {B8}

Defines the Canadian national keyboard standard for the English and French languages in text and office systems

5.1.5.2.2 Public Specifications

The specifications listed in this subclause are not part of the POSIX OSE, but may be of interest. Use of these specifications should be considered carefully because there are risks in using these unapproved specifications. They are included in this guide to indicate some of the existing work that has been done in areas that are gaps in the POSIX OSE.

- The X/Open XPG4 {B69} specifications.

5.1.5.3 Unaddressed Services

While the character set arena is heavily populated, very little work is being done in other areas of internationalization of products. Standards should be developed for

- Cultural conventions repository
- API services for cultural conventions
- API services for character set handling
- Multilingual collating rules
- Message delivery systems, including system messages
- Icon semantics for multinational applications
- Data announcement standards
- Configuration and administration of internationalized and localized objects

Additionally, no standards currently exist that support the following character set and data representation functionality:

Character set identifier

The application program should be able to write the character set identifier to data and should be able to retrieve the identifier for requested data.

Character set identifier comparison

The application platform should, upon request from the application or automatically, compare the character set identifiers of interacting data in the application (input, processing, data storage, communication, and output).

Character set translation

The application platform should provide translation of character sets, when requested by the application or automatically, when detecting a mismatch in the comparison process.

5.1.6 POSIX OSE Cross-Category Services

Not applicable.

5.1.7 Related Standards

Since it is a cross-component facility, internationalization affects nearly every element in information processing. Thus, almost all standards in this environment are related to the subject. The following are a few major families of standards strongly related to internationalization.

ISO DIS 8613 {28} ODA; see 4.5

ISO/IEC 8824 : 1990 {41}

ISO/IEC 8825 : 1990 {42} ASN.1; see 4.3

ISO/IEC 9945-2 : 1993 {69} Shell and utilities; see 4.7

Programming languages support local environments. See 4.1.

5.2 System Security Services**5.2.1 Overview and Rationale**

Information is the key to successful use of a system. For example, if used effectively and efficiently, information may be used to underpin enhanced service and to aid the derivation of strategic plans. Much of this information (for example, personal customer details and business financial plans) will be of a sensitive nature.

Although authorized users may be able to take advantage of the POSIX OSE to increase productivity and efficiency, unauthorized individuals may also be able to take advantage of the OSE to steal, manipulate, or deny others access to information held within the system, or to deny involvement in some transaction performed via the system. Similarly, even authorized users could unintentionally instigate some form of data corruption that inhibits system performance.

Therefore, security services have to be provided within the system if it is to prevent these unauthorized or accidental activities. To achieve an optimum degree of confidence in the correctness and effectiveness of the security services of a system, a system-specific security policy needs to be derived and appropriate security functionality designed into the system at the beginning of its life cycle.

A relatively high degree of protection for ordinary computer systems can be achieved if system administrators correctly configure and maintain the system according to recommended security guidelines and practice, such as those described within the X/Open Security Guide {B63} . However, additional security facilities have to be supported within the system to enhance protection against the small percentage of attackers who are determined to breach the security of the system. It is the intent of the security extensions to the base POSIX interface standard to support these additional security facilities.

The four basic security objectives of a system are to maintain

Confidentiality

The system has to prevent unauthorized viewing of data.

Integrity

The system has to prevent unauthorized alteration or deletion of data.

Availability

The system has to ensure that authorized users are not prevented from accessing and processing data.

Accountability

The system has to ensure that users are made accountable for their actions, and that audit trails for security enforcement are supported.

Different user groups may place different emphases upon these four basic security objectives. For example, the military security sector may place more importance upon confidentiality than accountability while, correspondingly, the commercial sector may place more importance upon accountability than confidentiality.

5.2.2 Scope

An effective, secure system provides defense in depth, such that if one layer of security is breached, then further layers of security will limit or prevent, or both, unauthorized or unintentional activities within the system.

To achieve a high degree of confidence in the correctness and effectiveness of the security of a system that will be processing sensitive information, security needs to be designed into the system at the beginning of its life cycle.

A System Security Policy (SSP) defines what it means for a specific system to be “secure” and, as such, forms the basic security input into the system life cycle. Specification of an SSP is therefore axiomatic to the design of a secure system.

Although the SSP defines what security measures will be provided within the system, it is the system design documentation that defines how these security measures will actually be implemented.

One aspect of an SSP may be that it mandates conformance with the POSIX security extensions.

IEEE P1003.6, one of the IEEE POSIX security groups, has responsibility for defining the security extensions (IEEE P1003.1e {B21} and IEEE P1003.2c {B23}) to the base POSIX interface standard. The POSIX security interface specifications are intended to assist in the construction of a secure system. They do not, in isolation, provide any protection against threats to a system.

5.2.3 Reference Model

The POSIX OSE reference model partitions the API and the EEL into a collection of services, each one of which may have a security component. These individual components may operate independently or cooperatively with other security components. For example, a user identification and authentication operation may include elements of the user interface services, communication services, and database services.

5.2.4 Services

Through an analysis of the potential threats and requirements of the system, the system security objectives, and hence, the necessary SSP rules, may be derived. This analysis also needs to take into account appropriate corporate, legal, and standardization requirements.

System confidentiality, integrity, availability, and accountability may be supported by the following functional security objectives:

Identification and authentication

A system entity, such as a user or system element, has to prove that its claimed identity is legitimate, such that another system entity may place confidence in that claimed identity.

Access control

Access to system resources has to be restricted to authorized entities only. Residual data contained within an object have to be securely erased before it may be reused by a system entity.

Accountability and audit

System users need to be made accountable for their actions. Audit trails of these actions then need to be maintained and utilized such that unauthorized system activity will be detected.

Accuracy

The system has to ensure that the correctness and consistency of security-relevant information is maintained.

Availability

System resources have to be provided to users in a consistent and reliable manner.

Data exchange

Data transmitted between system users or elements, or both, have to be protected from unauthorized interference or viewing. Originators and recipients of data have to be authenticated and have to be able to prove to one another their respective participation in the transaction.

Other security objectives are

Assurance

The security of the system has to be specified, designed, implemented, tested, and maintained in such a way that confidence can be placed in the correct and effective operation of the system. Also, procedures have to be specified to ensure continued confidence in the security of the system in the event that the system is modified in some manner.

Security roles and responsibilities

Security activities need to be able to be partitioned and allocated to identifiable security administrators, who will then be responsible for ensuring that their allocated task is satisfactorily performed.

Secure operating procedures

Procedures have to be written that will guide system administrators and users as to the correct procedure to follow in the event of some security-relevant occurrence.

5.2.4.1 API Services

The scope of the security interfaces is currently limited to extensions for those interfaces defined within ISO/IEC 9945-1 : 1990 {68}. Issues not addressed by the security interfaces include noninterface-specific architectural assurance issues and distributed or networked systems security. It should be noted that simply providing security services does not make a system “secure.”

5.2.4.1.1 POSIX Security Interfaces

The POSIX security interfaces include audit, privilege, discretionary access control (DAC), mandatory access control (MAC), and information labels (ILs). Implementation support for these POSIX security interfaces is optional.

Audit interfaces

These interfaces support the concept of accountability, wherein all system users are made accountable for their actions through the recording of user identities and associated actions within an audit trail. The audit interfaces support both portable audit generating and processing applications.

Privilege interfaces

These interfaces support the enforcement of a least privilege security policy, wherein system administration responsibilities and associated privileges are allocated to a number of discrete roles in order to compartmentalize the security impact of a subverted security administrator or of unauthorized usage of a security administrator role.

DAC interfaces

These interfaces support the imposition of a fine granularity of user specified access control to objects. The DAC interfaces supplement the user-group-other permission bits that currently exist within ISO/IEC 9945-1 : 1990 {68} systems through the use of Access Control Lists (ACLs).

MAC/IL interfaces

These interfaces support the imposition of system-specified policies for labeling. A MAC label is associated with an object (e.g., a file) and is typically used for access control purposes. An IL is associated with the data contained within the actual object itself and may therefore be used as a more accurate label for the purposes of hard-copy printing (e.g., a classification level label, such as “secret” or “confidential”). Further, an IL may contain information that is not related to logical access control and that is therefore not suitable for inclusion within a MAC label (e.g., physical document handling restrictions).

5.2.4.2 EEI Services

EEI Services are beyond the scope of the POSIX security interfaces defined in IEEE P1003.1e {B21} . However, distributed security services are being examined in the IEEE P1003.22 working group. EEI services would fall under their jurisdiction, and will be addressed in the forthcoming IEEE P1003.22 work. Some issues to consider include

- HCI
 - User login
 - Trusted path
 - User change password (or other authentication token)
 - Administration
- Communications/Data Interchange
 - Protocols for remote authentication and authorization, including syntax for subject and object security attributes
 - Security administration/management (e.g., password and key management)
 - Secure communications (integrity/dialogue protection)
 - Secure communications with non-OSE systems
 - UDB and ACL formats for interoperability

The user ID provided by the system to applications (even remote ones) has to be considered valid. Otherwise, security would be violated by the application requesting further identification in some of these environments.

5.2.5 Standards, Specifications, and Gaps

See Table 5.2.²¹

Table 5.3 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

Table 5.2—System Security Standards

Service	Type	Specification	Subclause
Confidentiality and integrity	E	IEEE P1003.1e {B21}	5.2.5.2.1
Access Control	E	ISO 8613 Draft Addendum {28}	5.2.5.2.1
Identification and authentication	S	ISO/IEC 9594-8 : 1990 {59}	5.2.5.1
Security roles and responsibilities	S	ECMA 138 {114}	5.2.5.1
Networked assurance	S	ISO/IEC7498-2 : 1989 {19}	5.2.5.1
Data exchange	G	n/a	5.2.5.3

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

²¹ Note that this list is not exhaustive. A summary of existing security standards activity can be found in section4, parts 1 and 2, of ISO/IEC JTC 1/SC21 N7292: “Revised Guide for Open Systems Security” (project 21.49 of ISO/IEC JTC 1/SC21/WG1). A more detailed description of specific ISO and ITU-T activities can be found in the “JTC 1 Catalogue of Security Work.”

Table 5.3—System Security Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
IEEE P1003.1e {B21}	E				E		

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
IEEE P1003.1e {B21}						

NOTES:

- 1 — LIS—Language-independent specification is available.
- 2 — Ada, APL, Basic,... —Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap
- 4 — All other standards shown in Table 5.2 have no APIs, and therefore no language bindings, so they are not applicable to this table.

5.2.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

ISO/IEC 9594-8 : 1990 {59}

Defines public key-based authentication standards for the X.500 directory service.

ECMA 138 {114}

Defines services and data definitions for security in open systems; complementary to ISO/IEC9594-8 : 1990 {59}.

ISO/IEC 7498-2 : 1989 {19}

Provides a general description of security services and related mechanisms, which may be provided by the OSI reference model, and defines the positions within the reference model where the services and mechanisms may be provided.

5.2.5.2 Additional Specifications**5.2.5.2.1 Emerging Standards**

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

5.2.5.2.1.1 IEEE P1003.6

The IEEE P1003.6 working group is addressing security extensions for those interfaces defined within the POSIX system API (ISO/IEC 9945-1 : 1990 {68} and ISO/IEC 9945-2 :1993 {69}). Issues not addressed within the IEEE P1003.6 scope include noninterface-specific architectural assurance issues and distributed or networked systems security. See IEEE P1003.1e {B21} and IEEE P1003.2c {B25} .

5.2.5.2.1.2 ODA Security

ISO 8613 {28} ODA Draft Addendum on Security. See 4.5.5.1.

5.2.5.2.2 Public Specifications

DOD 5200.28-STD {B9}

The US Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), colloquially known as the “Orange Book,” describes security requirements for trusted systems based on system specifications and data exposure. This standard consists of a number of interpretations for various system components (e.g., the Trusted Network Interpretation and the Trusted Database Interpretation.) These are referred to as the “Rainbow Books.”

ITSEC {B44} The Information Technology Security Evaluation Criteria is the European equivalent to DOD 5200.28-STD {B9} .

5.2.5.3 Unaddressed Services

The need for distributed security services has been identified, but has not yet been satisfied.

5.3 Systems Management Services

5.3.1 Overview and Rationale

Information systems are composed of a wide variety of diverse resources that have to be managed effectively to be successfully used. Although the individual resources may differ widely, the basic concepts of management can be applied in a uniform manner. The adoption of a common management model enables administrators to manage many resources without the need to learn different techniques for each resource.

The goal of system and network management standards is to enable the provision of portable and interoperable management technology. In addition, the existence of standard user-level interfaces to management functionality provides for the portability of system and network administrators.

An important element of standards in this area is that they should not enforce specific management policies. They should instead enable a wide variety of different management policies to be implemented, selected according to the particular needs of different end-user installations.

5.3.2 Scope

Systems management functionality may be divided up in several different ways. One scheme is to make a division according to the resources being managed. This will result in a series of functional groups such as

- Printer management
- Software management
- User management
- Network management
- Host management
- Processor management
- File system management
- Device management

This list is not exhaustive, but it is representative of the variety of resources that need to be managed.

It is also possible to identify a series of components that are common to all these functional groups. This results in a set of orthogonal categories that includes

- Configuration management
- Performance management

- Fault management
- Accounting management
- Security management

Systems management functionality may be accessible by means of APIs and CLIs as appropriate.

5.3.3 Reference Model

The reference model for systems management is consistent with the POSIX OSE model, as shown in Figure 3.3. Figure 5.1 illustrates management applications that provide software application entities to support management tasks in specific areas of management functionality. These management applications make use of the services provided by the application platform. In addition to the usual application platform services, there are management-specific components within the platform. These represent management system services that are required in order to implement management applications.

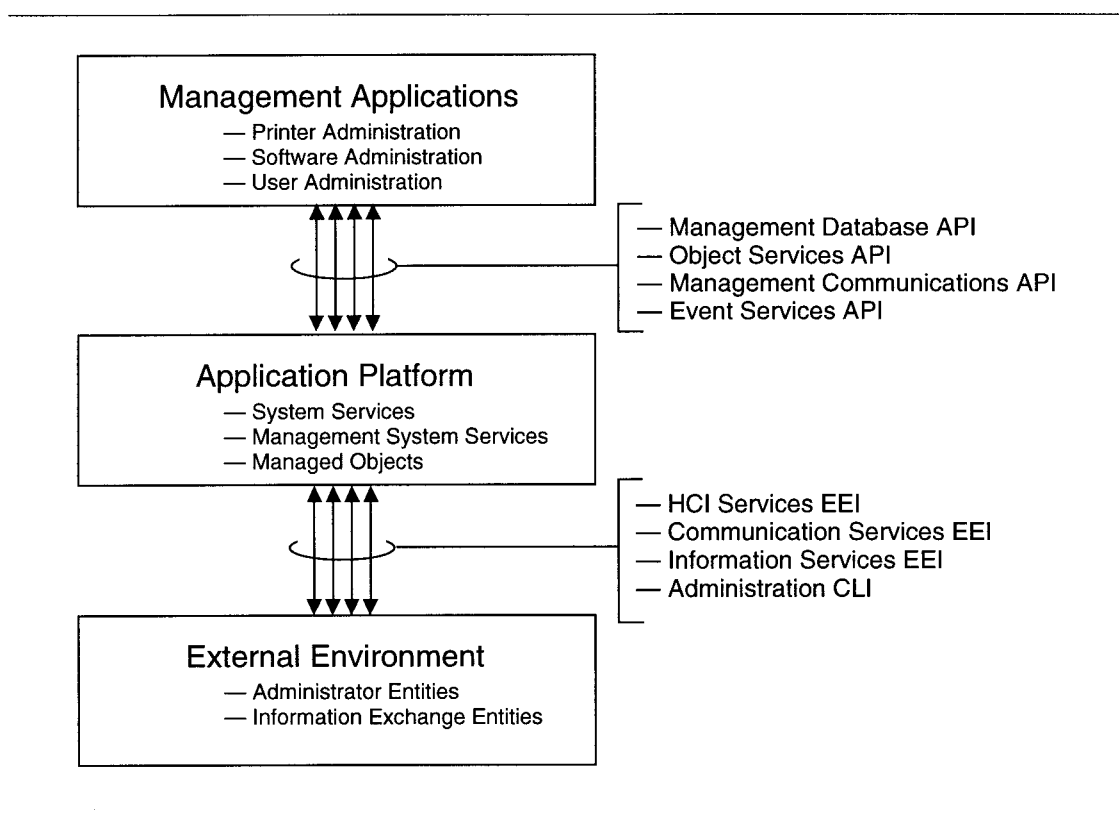


Figure 5.1—POSIX OSE Reference Model for Systems Management

The application platform also includes managed objects. Managed objects are abstract representations of resources that are to be managed. The managed object abstraction supports a well-defined methodology for accessing those properties of a resource that are to be managed. It is the use of this abstraction that allows widely differing resources to be managed in a uniform manner.

The external environment, as it is described for the systems management reference model, includes administrators who use management applications via a command line interface. Such command line interfaces (and graphical user interfaces) facilitate administrator portability. The external environment also includes means for exchanging

information outside the model. Such information exchanges can take place via mechanisms such as external networks and transportable media.

5.3.4 Services

This subclause identifies the services necessary to provide effective control and management of information technology resources. An enterprise will assess the level and quality of information technology services needed to match its business needs and user requirements, considering issues such as service initialization, configuration planning and management, software installation and maintenance, performance monitoring, and problem resolution.

5.3.4.1 API Services

While the user of these services, e.g., the operations or service manager, will access them across the EEI, a number of the services will be supported in turn by software entities on the platform, such as a database to support configuration management. These underlying services with their associated APIs will be readily identifiable with the fundamental platform services described in Section 4. As efforts are made to standardize the following services, additional APIs are being identified that will require standardization. The following subclauses describe the characteristics of several different management services. This guide does not attempt to provide a definitive list of all possible such management services, and the following subclauses provide descriptions only for a representative set of such management services.

5.3.4.2 Configuration Management

Configuration management consists of four basic functions: identification, control, status accounting, and verification.

Identification involves specifying and identifying all of the component resources of an information system that need to be managed. These configurable items include software, peripherals, magnetic media, network services, distributed services, etc.

Control implies the ability to agree and “freeze” configuration items (CIs) and then to make changes only with agreement of the appropriate named authorities. Control is concerned with ensuring that none of the CIs shown is altered or replaced and that no CIs are added without appropriate authorization.

Status accounting involves the recording and reporting of all current and historical data concerned with each CI. This may include maintaining records of the current, previous, and planned states and attributes of the CIs and tracking these states and attributes; for example, as the status of a CI changes from “development” through to “test,” “scheduled to go live,” “live,” and through to “archived.”

Verification consists of a series of reviews and audits to ensure that there is conformity between all CIs and the authorized state of CIs as recorded in the configuration management database (CMDB). It is concerned with checking that the physical CIs actually match the authorized system as described in the CMDB.

5.3.4.3 Software Management

The main types of software to be installed and distributed are application programs developed in-house, purchased applications, and utility software. All software needs to be managed effectively from development or purchase through to the live environment. Unless the distribution and implementation process can be controlled automatically, or remotely controlled from a central management facility using software tools, procedures have to be in place to ensure that distributed software arrives when expected, is checked for authenticity and to the required level, and that the software is brought into use when required. The main requirements for services include

- Control of installation
- Control of distribution
- Configuration assessment
- Configuration verification

5.3.4.4 System Initialization, Reinitialization, and Shutdown Services

System initialization consists of services for a complete restarting of the software, starting up the attached hardware subsystems devices, doing subsystem and system self tests, and completely initializing the database.

System reinitialization consists of services for restarting the software while using the existing database information. The software may have to be reloaded, and the database may have been reestablished by a system recovery. Attached hardware subsystems may also need to be reinitialized.

Reinitialization also includes a function to restart applications redistributed to other processors after a processor module failure.

Shutdown services are those required to perform planned orderly shutdowns at the local and remote levels for each and all processor(s) throughout a system. These services support both crisis and noncrisis situations that call for system shutdown. They make sure that the persistent store is in a consistent state; see to the clean termination of all processes, programs, devices, etc.; and take care of user notification. They also provide for the running of system diagnostics.

5.3.4.5 License Services

The terms and conditions relating to the supply of software may place legal restrictions on the organization (e.g., no unauthorized copies to be made). The service would allow only a predefined number of copies to be running simultaneously. License management is an emerging management area, which is important for two reasons. Effective enforcement of license agreements allows users to negotiate individual licenses with vendors that cost-effectively meet their needs. This assists the organization in discharging its legal obligations and assists auditors in checking for the existence of unauthorized copies.

All authorized copies of licensed or purchased software that are made by system management staff should be allocated a unique copy number and recorded in the configuration management database, together with their location and who is responsible for them. Procedural restrictions should be introduced to prohibit the unauthorized copying of software, and regular software audits should include a check for any unauthorized copies.

5.3.4.6 Print Output and Distribution Services

Output and distribution packages control output production and distribution from the moment the output is planned to the time the user receives the print. Working criteria need to be established first; e.g., define who receives the output and how much of it the user gets.

The main functions are

- The printed output can be limited to parts wanted by the user
- Multiple copies of the entire output, or of selected sections, can be produced
- Output can be grouped by recipient within delivery location
- Output for each job can be spooled as a group when the job is complete
- The number of whole printouts and individual pages received by each user can be recorded
- Output production can be monitored and managed efficiently

Output and distribution packages should include the following:

- Printing and distribution of whole and partial printouts
- Status (queued, printing, etc.) of the output tracked
- Online viewing of text to be printed
- Archiving of output files
- Support for a wide range of printers
- Costing and charging functionality
- Security facilities

By using an output distribution package, the delivery of printouts to the correct person at the correct location can be ensured. Paper, time, and information technology resources are saved as the users receive only the parts of reports that they need. They can also view the text online. The number of pages printed can be controlled. Output can be tracked from the time it is created to the time it is delivered to the user, allowing for good security monitoring.

5.3.4.7 Office Media Management and Backup/Restore

The main services of magnetic tape and data cartridge management systems are to

- Provide automated support for tape housekeeping and maintenance, including
 - Allocating tapes and releasing them for reuse
 - Ensuring even patterns of use (i.e., the same tape is not continually reused)
 - Constructing and triggering cleaning schedules
 - Maintaining the security of data
- Automate archiving (vault management) for offsite storage
- Identify growth requirements

Vault management is concerned with controlling the movement of tape cycles (i.e., generations of backup, such as grandfather, father, son) from one storage location to another. As a tape cycle is used, the tape management system automatically logs a different vault identifier against each tape.

A backup strategy is required to control the frequency of backups and the way in which they are created; e.g., whole volumes to cartridge or individual files to tape.

The backups and restores of system and application software should be separate from the backups and restores of data. Software and library backups should be explicitly scheduled and the complete software item or library backed up. The schedule for backing up files needs to be fully documented, properly maintained, and adequately safeguarded as the contents of the schedule are required for disaster recovery purposes.

5.3.4.8 Online Disk Management

The operation of disk management systems requires that they take account of a range of factors such as retention period, recovery, space fragmentation, disk overflow, file and record activity levels, and channel use. Some systems merely report against values or thresholds set, but increasingly they invoke corrective action. Typically, the corrective action is file and disk reorganization or file and data archiving.

If a disk management system is used, the constant monitoring and actioning of requests for disk space can be minimized. Disk space may be collectively pooled and unused space constantly reclaimed.

5.3.4.9 Job Scheduling

Scheduling involves the continuous organization of jobs and processes into the most efficient sequence, maximizing throughput and utilization to meet the targets set in service level agreements. Jobs are scheduled to ensure

- Service level agreements and user requirements are met; e.g., certain jobs need to be initiated and completed by a certain time
- Available capacity is used effectively; e.g., the workload run at any given time does not exceed the practical capacity

The minimum services of a scheduler should include

- A high upper limit for the number of relationships allowed between jobs
- The ability to schedule by calendar and criteria
- Workload balancing support

- Levels of security
- Ability to restart jobs
- Operator override capability
- Capability to model future workloads

5.3.4.10 User Administration

User administration services should provide the ability to

- Create a new user or group of users
- Delete a user or group of users
- Allocate system resources to a user or a group of users

5.3.4.11 Accounting

An effective accounting system allows an organization to track usage of computing resources by various users and groups. The typical purpose of accounting services is to charge for services and allocate resources among users and groups of users.

Accounting services include the ability to

- Measure and prioritize resource usage
- Report costs and usage to management and users
- Establish resource limits for users and groups of users
- Enforce resource limits for users and groups (i.e., when resource limits have been exceeded, access is prohibited)

5.3.4.12 Performance Management

Efficient management of the information technology resources requires, among other things, that performance aspects of hardware, software, and network components be monitored and that there are services and parameters to provide the system manager with the ability to tune the system to meet performance targets. Of course, there will be specialized applications that may not need such management services.

Items to be monitored for the host hardware include

- CPU idle time
- Total CPU utilization
- I/O subsystem
- Virtual memory system
- Average main memory occupancy

Software aspects to be monitored include

- Start and finish times
- Total measured CPU utilization
- Total number of physical I/Os
- Average (and maximum) occupancy of main memory
- Paging and swapping rates
- Amount of disk space allocated or deleted
- I/O by device
- Transaction data and utilization statistics for TP and database systems

Items to be monitored on the network include

- Total number of messages sent and received
- Total bytes sent and received
- Number of polls
- Number of messages retransmitted
- Utilization of controller/line
- Average response time

5.3.4.13 Capacity Management

An effective and efficient capacity management function contains at least the following elements:

- Performance management to monitor and optimize the use of current systems.
- A capacity management database that contains current and historic data of technical and business-related interest. This database forms the basis for the provision of both tactical and strategic reports on performance and capacity.
- Workload management to identify and understand the applications that use the system. The understanding of workloads has both a technical and business-related nature. This involves application sizing to predict the performance and required capacity of new applications accurately.
- Capacity planning to plan accurately the required hardware resource and associated cost for the future and to predict the effect on performance and capacity of both tactical and strategic plans.

5.3.4.14 Fault Management

Fault management services allow a system to minimize the impact of system faults. There are a number of areas of fault management.

5.3.4.14.1 Status of System Components

Maintainability services provide support for the maintenance of a system. A major component of that support is the collection and logging of information about the operation of the system. Typical information to be logged is

- Software and hardware errors during operation
- Processes that failed or almost failed to meet scheduled deadlines
- Performance metrics for system tuning
- Times when the system operated in extreme environmental conditions
- Errors reported during startup self-testing
- Attempts to violate rules of the security policy of the system

5.3.4.14.2 Fault Avoidance

These services involve the avoidance of faults before a failure in the system component occurs. If a system can detect that the operation of a component is approaching the edge of its operational range, a standby or backup component could be phased in to replace it. Another form of fault avoidance is logging of shocks, temperature extremes, etc., so that it can be predicted that a component will not meet its original expected service life.

5.3.4.14.3 Fault Detection

Fault detection services are concerned with determining when a fault has occurred in the system. Fault detection services are both passive and active. Active services are those that attempt to determine the status of various system components by testing those components. Passive services, on the other hand, try to ascertain system components by passively gathering information and watching the behavior of the system.

5.3.4.14.4 Fault Isolation

Fault isolation services attempt to determine the component at fault and segregate the faulty component from the rest of the system. Services may be shared between the fault detection and isolation service library in that they perform both functions.

5.3.4.14.5 Fault Recovery

Fault recovery services attempt to bring the system into a consistent state. These services may have an impact on scheduling services, network services, and database services, depending on the recovery scheme used.

Redundancy of resources is often needed to support fault recovery. Resources may include data, process, processor, disk drive, etc.

As parts of the system fail, it may no longer be possible to satisfy all the requirements of the application. Services to support graceful degradation may be used to ensure that critical activities do not fail.

5.3.4.14.6 Reconfiguration

These services allow the system to reconfigure its view of the world and to substitute different resources to perform system functions such as substituting a new physical I/O channel to support a logical channel. These services are part of the API, but their use may be restricted to specially authorized programs such as those used by the target system operator.

5.3.4.15 Security Management

Systems management should exploit the security services to provide secure delivery and installation. Any changes to a system should be audited, and only authorized subjects should be able to make changes to a system. Security policy should be administrable across the network.

5.3.4.16 EEI Services

System administrators and other users need a command-line user interface or a graphical user interface, or both, in order to access and use the system management services. Many of the supporting services described in 4.7 and 4.9 are applicable in this respect.

5.3.5 Standards, Specifications, and Gaps

See Table 5.4.

Table 5.5 summarizes the applicability of the various standards to the various programming languages supported by the POSIX OSE.

5.3.5.1 Standards in the POSIX OSE

The standards listed in this subclause are part of the POSIX OSE.

5.3.5.1.1 Performance Management Standards

ANSI X3.102-1992 {105} defines parameters that can be used to describe the performance of data communication services. Parameters describe performance between pairs of users and apply to all classes of data communication systems and services.

Table 5.4—System Management Standards

Service	Type	Specification	Subclause
Configuration management	S	ECMA TR-47 {113}	5.3.5.1
Software management	S	IEEE Std 1387.2-1995 {131}	5.3.5.1
System initialization/ reinitialization/shutdown	S P	ISO/IEC 9945-1 :1990 {68} XPG4 {B69} , OSF AES-OSC {B50}	5.3.5.1 5.3.5.2.2
License services	G	n/a	5.3.5.3
Print services	E	IEEE P1387.4 {B33}	5.3.5.2.1
Office media management	G	n/a	5.3.5.3
Online disk management	G	n/a	5.3.5.3
Job scheduling	G	n/a	5.3.5.3
User administration	E	IEEE P1387.3 {B32}	5.3.5.2.1
Accounting	S	ISO/IEC DIS 10164-10 {78}	5.3.5.1
Performance management	S	ANSI X3.102-1992 {105}, ANSI X3.141-1987 {108}	5.3.5.1
Capacity management	G	n/a	5.3.5.3
Fault management	S	ISO/IEC 10164-4 :1992 {74}, ISO/IEC 10164-5 :1993 {75}, ISO/IEC 10164-6 :1993 {76}, ISO/IEC 10164-7 :1992 {77}	5.3.5.2.1
Security management	G	n/a	5.3.5.3

NOTE — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

Table 5.5—Systems Management Standard Language Bindings

Standard	LIS	Ada	APL	Basic	C	C++	COBOL
ISO/IEC 9945-1 :1990 {68}	E	E			S		
IEEE Std 1387.2-1995 {131}	E				S		
IEEE P1387.3 {B32}	E				E		
IEEE P1387.4 {B33}	E				E		
ANSI X3.102-1992 {105}							
ANSI X3.141-1987 {108}							

Standard	C-Lisp	Fortran	Modula-2	Pascal	PL/I	Prolog
ISO/IEC9945-1 :1990 {68}		E	E			
IEEE Std 1387.2-1995 {131}						
IEEE P1387.3 {B32}						
IEEE P1387.4 {B33}						
ANSI X3.102-1992 {105}						
ANSI X3.141-1987 {108}						

NOTES:

- 1 — LIS — Language-independent specification is available.
- 2 — Ada, APL, Basic, ... — Language-dependent specification is available.
- 3 — S, E, P, G — Standard, Emerging Standard, Public Specification, Gap

ANSI X3.141-1987 {108} provides uniform general methods of measuring values for the parameters described in ANSI X3.102-1992 {105}. They may be used in measuring performance values at any pair of digital interfaces connecting a data communication system to its users.

5.3.5.1.2 Configuration Management Standards

ECMA TR-47 {113} addresses the requirements for configuration management from the perspective of the system manager. The services defined are applicable to any situation in which configuration management information is to be transferred.

5.3.5.1.3 POSIX Software Administration

IEEE Std 1387.2-1995 {131} defines a software packaging layout, a set of information maintained about software, and a set of utility programs to manipulate that software and information.

5.3.5.2 Additional Specifications

5.3.5.2.1 Emerging Standards

The specifications listed in this subclause are not part of the POSIX OSE, but they address services included in this guide and will qualify for inclusion in the POSIX OSE upon formal approval by their parent standards development organizations. Use of these specifications should be considered carefully as there may be risks in using these emerging standards prior to their final approval.

5.3.5.2.1.1 POSIX System Administration

The initial scope for the IEEE P1387 system administration project produced a long list of topics for consideration. In addition to IEEE Std 1387.2-1995 {131}, the following drafts are in development:

IEEE P1387.3 {B32}	User management
IEEE P1387.4 {B33}	Print management

5.3.5.2.1.2 OSI Systems Management

The following is a representative subset of OSI systems management projects that are approved or under development in SC21:

ISO/IEC 10164-4 :1992 {74}	Alarm report function
ISO/IEC 10164-5 :1993 {75}	Event report management function
ISO/IEC 10164-6 :1993 {76}	Log control function
ISO/IEC 10164-7 :1992 {77}	Security alarm reporting function
ISO/IEC DIS 10164-10 {78}	Accounting meter function

5.3.5.2.2 Public Specifications

The X Management Protocol (XMPP) {B57} specification addresses the Common Management Information Protocol (CMIP) [Open System Interconnect (OSI)] and the Simple Management Network Protocol (SNMP) [Internet Protocol Suite (IPS)].

5.3.5.3 Unaddressed Services

The following services currently have no standards work in progress:

- License services

- Office media management
- Online disk management
- Job scheduling
- Capacity management
- Security management

5.3.6 POSIX OSE Cross-Category Services

- Security for remote print jobs
- Distributed management communications
- Security for distributed management communications

5.3.7 Related Standards

None.

6. Profiles

Profiles are a response to the growing confusion originating from the ever-expanding inventory of nonrelated and overlapping individual standards. One of the major motivations behind the profiling activities is to simplify the procurement work that has to be done by users who want to specify an OSE and by developers who want to build portable applications that need a combination of standards.

There are two audiences for this section:

- People who want to know more about profiles
- People who are in the process of developing their own profiles, such as those developing formal SPs (see the definition of standardized profile in 2.2.2.50) and those developing less formal profiles for their industry group (e.g., a banking trade association) or enterprise for the purpose of procurement or strategic planning

NOTE — Those intending to develop profiles for standardization within ISO should consult ISO/IEC TR 10000 {B1} . Guidance specific to OSE profiles is provided in TR 10000-3.

The concepts in 6.2 should be useful for both audiences. Guidance is offered in 6.3 for those who write profiles, but useful definitions of terms are also given to explain concepts more precisely. The different types of profiles are described in 6.4. This classification is not yet based on a large number of existing profiles since the profiling activity is relatively new in the POSIX environment. In that respect, this classification may evolve.

6.1 Scope

The information presented in this section is limited in scope to assist those needing to understand profile concepts as they apply to the POSIX OSE. Profiles that are constructed from base standards (and profiles) listed within this guide (see 1.2) are covered by this section.

Although certain aspects dealing with profiles (including profile management, profile conflict resolution, profile update, and profile verification and validation) are very important, they are not within the scope of this guide. These aspects are left to the standards development organizations.

6.2 Concepts Related to Profiles

6.2.1 Introduction

This guide is designed to assist in the selection of standards in the procurement process or as a target application environment. Existing profiles also assist in the selection of standards for a particular purpose by providing a place to record standards that successfully supported that purpose in the past. This accumulated experience saves time and expense of duplicated searching of available specifications. A profile is a suite of base standards with specified options. Profiles can be created by software developers to describe the environment they target, or by buyers to identify their purchasing objectives.

Profiles can provide a clear definition of how one or more standards (and other profiles) fulfill a set of functional requirements.

6.2.2 Basic Terminology

This subclause introduces the relationships between profiles and base standards.

There are two general classes of standards documents:

- Base standards
- Profiles, including AEPs, SPs, and POSIX SPs

Base standards specify functionality, syntax, protocols, data formats, etc., in detail. Profiles (sometimes called functional standards) identify which base standards are applicable. Base standards often consist of a base or mandatory part (not necessarily the bulk of the base standard) and a number of optional parts and values. Profiles may (or may not) choose specific options or values for each base standard. A profile may also identify other profiles, allowing the construction of profiles built from both base standards and other profiles. If an organization is developing a profile that is not intended to be a formal SP, the organization may have more flexibility in the choice of base standards for that profile. The relationships between profiles developed within the formal standards process and profiles developed within user organizations is addressed more specifically in 6.4.

NOTE — In the context of internationalization, the term “national profile” is frequently used and will be found, for example, in ISO/IEC 9945-1 :1990 {68} and ISO/IEC 9945-2 :1993 {69}. Its meaning is consistent with the definitions in 2.2.2, but in many cases such profiles reflect national cultural conventions. For example, Denmark and Japan both have specified national profiles dealing with character sets. Any of the profile types described in 6.4 may be considered “national profiles” if they include standards associated with national cultural conventions.

6.2.3 Relationships Between This Guide and Profiles

The relationships that exist between profiles, this guide, and the base standards is a key to understanding profiles. The base standards listed in 1.2 form the basis of the POSIX OSE. The profile developer selects the base standards needed to provide portability and interoperability in a given functional area. In the process, the profile writer grapples with the coherence of the selected base standards by choosing only those that will work together to get the particular job done. Profile writers should also deal with *harmonization* (see 6.3.2.4), which means making the profiles consistent with each other where they overlap. This can often be done among profiles even where the functional areas served differ greatly. Procurements specifying two profiles that have been harmonized by their authors have the benefit of knowing that the two will not conflict with each other where they overlap.

If it is appropriate, a profile may reference another profile, building upon its definitions and interactions. This should make constructing the new profile easier and means that it is already harmonized with the first, rather than repeating the material. Care should be taken here: just as the first profile may be affected by changes in the base standards, the later profile may also be affected by changes in the profiles upon which it is based.

NOTE — An analogy is in the construction of electronic equipment such as computers. The basic building blocks are “components,” such as memory chips and capacitors, which can be fabricated into larger building blocks such as printed circuit boards, which can be fabricated (with other components or printed circuit boards) into larger building blocks, such as stand-alone computers, which can be fabricated into larger building blocks such as department-wide networks of computers, etc. Likewise, a few base standards (the basic building blocks), can be gathered together into profiles of components, which can then be gathered together (with other base standards or profiles of components) into larger platform profiles, which can be gathered together into larger AEPs. (See 6.4.)

Profiles are important for a number of reasons:

- Profiles provide a concise statement of specifications that describe the standards for the target functional objective(s).
- Profiles include information about the relationship between the selected standards (i.e., coherency is an objective).
- Profiles may identify the specific standards needed for an application domain, and therefore can be used in procurement or as a target for applications development.

6.3 Guidance to Profile Developers

This clause expands the concept of profiling in the manner needed by profile writers and provides detailed guidance to those writers. It includes a description of the basis for this guidance, expands on the purposes served by profiles, and finishes with more detailed guidance specifically aimed at those writing profiles.

Using this guide as a basis, profile writers can develop their own informal profiles, suited to their own needs, or formal standards bodies can develop formal, balloted profiles. This clause details the requirements that should be considered by developers of profiles whether they are POSIX SPs, SPs, or less formal profiles. SPs are formal profiles in that they meet the criteria of a standards body. SPs that also meet the requirements for POSIX-based profiles (rules established by the IEEE) are called POSIX SPs.

Some of the ideas and concepts for profiling described in this section derive from the work of ISO/IEC JTC 1 SGFS as documented in ISO/IEC TR 10000 {B1} , particularly profile content and format guidance in 6.3.2.3. Additionally, some consideration was given in this guidance, above and beyond ISO/IEC TR 10000 {B1} , to addressing the following:

- SPs
- POSIX SPs as conceptual extensions to ISPs

Developers of profiles following the guidance of this section should refer to the appropriate IEEE guideline documents, such as the PASC Profile Steering Committee Guidelines {B35} , if they intend to propose IEEE acceptance as a POSIX SP, and to ISO/IEC TR 10000 {B1} , if they intend to propose acceptance as an ISP.

6.3.1 Purpose of Profiles

Profiles define combinations of base standards and profiles for the purpose of

- Identifying the base standards, together with appropriate classes, subsets, options, and parameters that are necessary to accomplish identified functions for purposes such as interoperability and portability.
- Providing a system of referencing the various uses of base standards that is meaningful to both users and relevant suppliers
- Enhancing the availability for procurement of consistent implementations of functionally defined groups of base standards that are expected to be the major components of real application systems

6.3.2 Detailed Guidance to Profile Developers

6.3.2.1 The Relationship to Base Standards

Base standards specify procedures and formats that facilitate application portability and interoperability. They provide options, anticipating the needs of a variety of applications and taking into account different capabilities of real systems and networks.

Profiles further promote portability and interoperability by defining how to use a combination of base standards for a given function or application area. Profiles, by definition, do not define new application interfaces.

In addition to the selection of base standards, a choice may be made of permitted options for each base standard and of suitable values for parameters left unspecified in the base standard.

Profiles do not contradict base standards, but make specific choices where options and ranges of values are available. Profiles include all of the items made mandatory by the standard. The choice of the base standard options should be restricted so as to maximize the probability of interworking between systems implementing different selections of such profile options, consistent with achieving the objectives of the profile.

A profile makes explicit the relationships between a set of base standards used together (relationships that are implicit in the definitions of the base documents themselves) and may also specify particular details of each base standard being used.

6.3.2.2 Conformance Requirements

A profile may contain conformance requirements that are more specific than those of the base standards to which it refers. Conformance to a profile implies, by definition, conformance only to the selected options of the base standard(s) that the profile cites as normative references. Conformance to the base standard(s) does not necessarily imply conformance to the profile, nor does conformance to the profile necessarily imply conformance to the entirety of any base standard.

6.3.2.3 Main Elements of a Profile

A profile should comprise the following elements:

- Scope and purpose of the profile, which should be described in terms of which user needs are intended to be met by the profile
- Reference to a set of base standards and other profiles, including precise identification of the actual texts of the base standards and profiles being used and of any approved amendments and technical errata or corrigenda, conformance to which is identified as having a potential impact on achieving portability and interoperation using the profile
- Specification of each base standard and profile referenced, including the selection of options, ranges of parameter values, and other choices allowed by the base standard(s) or profile(s) referenced therein
- Statements defining the requirements to be observed by systems claiming conformance to this profile
- Definitions for any new concepts or terms introduced in the profile and not defined in the base standards or profiles being used

Informal profiles could include examples to clarify the intentions of profile writers for profile users; however, tutorial examples are not appropriate for ISPs.

Systems that interoperate can perform different but complementary roles (e.g., an initiator-responder or a master-slave relationship). In such a situation, the profile should identify the separate roles that may be adopted by a system, and these should be stated as either mandatory requirements or options of the profile, as appropriate.

6.3.2.4 Profile Objectives

The following objectives need to be considered in the writing of a profile; however, they do not suggest any document organization other than the one outlined by the profile definition document in the previous subclause.

6.3.2.4.1 Completeness

A profile should be complete with respect to its functionality objectives. This may well be an iterative process, since the understanding of the requirements and standards will evolve. Completeness means that all areas where standards should be applied have been identified and the requirements defined. Where standards exist, they have been included, and the options within those standards have been addressed. Where standards do not exist, but are needed, this has been documented in the profile. (When gaps of this type are identified, it is suggested that a statement of the requirements and priority for such work be forwarded to the appropriate standards body.) However, the profile has to be an engineering specification of standards (which means potentially testable) and has to be useful to applications developers and procurement organizations.

It may be appropriate to document (in an informative annex) specifications and alternatives available in areas where standards have not been defined. The meaning of this concept will be relative to the forum for acceptance of the profile. If the profile is targeted at ISO acceptance, then ISO DIS and IS standards should be the reference point, whereas a US Government profile might be focused on FIPS, ISO, or ANSI standards. Within private industry, consortium- and even vendor-specific specifications could be incorporated, keeping these as examples and not explicit requirements, which will simplify harmonization with formal standards as they emerge. Where standardized profiles are being developed and gaps are identified, the profile writer should identify the requirements that are not satisfied by a standard. If there is a preliminary specification available that addresses many of the requirements, that specification should be referred to informatively.

6.3.2.4.2 Clear Communications

A key objective for the profile is clear communications between the affected parties. Users, software developers, and platform suppliers all need to have the same terms and specifications. The application software developers and system vendors need a common set of specifications to target for their development efforts.

6.3.2.4.3 Harmonization

Harmonization²² can often be done among profiles even where the functional areas served differ greatly. This assures that the maximum practical agreement exists between different profiles, maximizing the implementations of that common ground. Harmonization is essential to permit a platform to provide multiple profiles.

6.3.2.4.4 Coherence

The simple selection of a group of standards does not assure that they will work together on a platform in a predictable way. A profile should contain a statement of how each standard cited as a normative reference interacts with the others, where applicable. Explicitly describing how such standards should not interact is also appropriate. Care should be taken when one of the referenced documents is itself a profile to ensure that interactions are made explicit, where necessary, between standards in the primary profile being defined and standards in the referenced profile.

6.3.2.4.5 Gap Identification

In the process of developing profiles, there may be gaps in coverage by standards that become apparent. These may exist in terms of the characteristics available with one standard that need to be made available from another, missing standards, or additional functionality that is needed for a specific applications activity. Therefore, an additional

²² This term, defined in 2.2.2.16, should not be confused with *international harmonization*, which refers to a specific process that needs to be followed in the approval process for ISPs.

objective for a profile effort is to document the requirements for such additional work and forward it to the appropriate standards effort. Profile groups in industry should consider providing expertise to the associated standards groups to assure that the resulting standards meet the needs of that application area.

6.3.2.4.6 Conformance

A profile addresses conformance. In the simplest case, the conformance section of a profile points to the conformance requirements of each of its base standards or authorized subsets. In more complex cases, additional conformance requirements will be necessary to govern the interactions among the base standards.

6.3.2.5 Outline of a Profile Development Method

Development of a profile is accomplished through the method of mapping out requirements, followed by the correlation of these requirements to standards. The purpose of this subclause is to discuss topics that such a method would include.

User requirements

A precise definition of the detailed requirements as a list of functions and a list of attributes and architectural constraints is required. This is a prerequisite for the other steps, in the sense that it forms the rationale for the selection of standards. User requirements may refer to already existing profiles.

Profile architecture

A profile architecture addresses how the different components of the profile are combined.

Profile specification

Corresponding to the user requirements and the profile architecture, the specific profiles or standards are referenced. The technical specification of the interfaces between the components has to be resolved. These technical specifications may be of different classes, such as HCI, format, programmatic interface, or protocols.

6.4 Types of Profiles

Various groups and organizations have already begun the process of developing profiles appropriate for their particular instance of an OSE. While profiles for OSI have matured over a period of time and have been typed according to a rather precise taxonomy, the same is not yet true for OSE profiles. Consequently, the profiles emerging for OSE fall into more general types or categories. The terminology and definitions of these categories will continue to evolve as more experience is gained in the definition and use of OSE profiles.

A “single-standard” profile (such as FIPS Publication 151-2 {B18}) may consist of a subset of a particular standard or a single standard where parameters and options have been selected. This type of profile is often used when there is a wide range of options and parameters in a base standard and specifying these options can focus implementation efforts. It is important to be aware that some base standards reference other base standards normatively even when defining a single-standard profile.

A “platform profile” may consist of a combination of multiple standards, as well as single-standard profiles, representing a useful building block, which can be used to support a wide range of applications. The scope of a platform profile might be equivalent to what is generally thought of as an “operating environment.”

An AEP may consist of a combination of multiple standards, as well as single-standard profiles or platform profiles, or both, that specify diverse types of functionality needed for a particular application environment. Examples include desktop operations, accounting, and inventory control. The Petrotechnical Open Software Corporation (POSC) is defining AEPs to serve as the common infrastructure supporting their technical applications, which address the typical Exploration and Production (E & P) business requirements (see B.3).

User organizations can take advantage of the profile concept and related activities. This can include working in the formal processes to ensure that the essential single-standard profiles, platform profiles, and AEPs are defined. Industry organizations (such as POSC) are also defining application environment profiles for specific areas. Such profiles can yield a broader base of common implementations and applications. However, many applications are industry and organization independent. For example, the applications that most users need at the desktop (including word processing, electronic mail, spreadsheet, calendar) are universal. As such, specific industries may include these more generic AEPs into their own profiles.

In the short term, organizations will need to define profiles today without the benefit of standardized or industry-specific profiles. Groups creating their organization-specific profiles today should plan on evolving to the more broadly accepted industry profiles as these profiles are developed and mature.

7. POSIX SP Profiling Efforts

7.1 Introduction

Profiles are not new; in the past, profiles were developed in companies and organizations. They were called “corporate guidelines” or “procurement guidelines,” and they specified mainly proprietary products. An SP differs from these past profiles in that it refers to one or more base standards. Where applicable, it identifies chosen classes, subsets, options, and parameters of these base standards in order to accomplish a particular function. A POSIX profile is an SP that also specifies certain POSIX base standards to support a class of applications or platforms. In addition, to be a POSIX profile, the profile cannot depart from the reference model defined by this guide and has to be harmonized with other POSIX profiles.

This section lists the POSIX SPs. It describes which POSIX SPs are approved or in preparation at the time this guide was approved, together with a summary description of the scope, scenario, and model for each profile. These POSIX SPs might be useful as building blocks for other profiles. The POSIX SPs include both AEPs and platform profiles.

7.1.1 Approved POSIX SPs

At the time of approval of this guide, there were no approved POSIX SPs.²³

7.1.2 POSIX SPs in Progress

The current efforts to develop POSIX SPs are summarized in Table 7.1.

²³ A *Standards Status Report* that lists all current IEEE Computer Society standards projects, including POSIX SPs, is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903, USA; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

Table 7.1—POSIX SPs in Progress

Project Name	Profile Subject	Profile Type
IEEE Std 1003.10-1995 {121}	Supercomputing	AEP
IEEEP1003.13 {B27}	Minimal (Embedded) Realtime System Profile	AEP
IEEEP1003.13 {B27}	Realtime Controller System Profile	AEP
IEEEP1003.13 {B27}	Dedicated (Mid-Range) Realtime System Profile	AEP
IEEEP1003.13 {B27}	Multipurpose (High-End) Realtime System Profile	AEP
IEEEP1003.14 {B28}	Multiprocessing Application Support	Platform profile
IEEEP1003.18 {B29}	POSIX Interactive Systems AEP	AEP

NOTE — At the time of approved of this guide, it was not known whether the four realtime profiles would be contained within a single multipart POSIX SP or separate single-part POSIX SPs.

7.2 Multiprocessing Systems Platform Profiles

The POSIX Multiprocessing Systems Profile (IEEE P1003.14 {B28}) is a platform profile. Like the POSIX Interactive Systems AEP (IEEE P1003.18 {B29}), the multiprocessing systems profile defines the functionality, standards, and options within standards that are needed for development and execution on a multiprocessing platform.

The multiprocessing systems profile is intended for use by multiprocessor vendors, application developers, users, and system administrators. It is important because it is designed to support the portability of multiprocessing applications, users, and system administrators in multiprocessing environments.

The multiprocessing systems profile has two major goals. The first one is to make POSIX systems reliable for multiprocessing. This goal requires IEEE P1003.14 to identify and address the caveats, problems, and limitations of POSIX base standards for multiprocessing platforms. Examples of these problems and limitations range from reentrant-function problems to potential problems with threads.

The second goal is to make POSIX systems useful for multiprocessing. This goal requires IEEE P1003.14 to ensure that POSIX systems support the functionality needed by multiprocessing platforms. An example of this is ensuring that POSIX systems have capabilities to allow vendors to implement software functions so the functions can be executed in parallel. In the absence of parallelizing standards, the details of what happens when the same software functions are used on different multiprocessor system vary.

The multiprocessing systems platform profile identifies standards, options, and gaps in the standards relevant to multiprocessing. It also identifies additional requirements not satisfied by existing standards and, in an informative annex, suggests interfaces to extended services that can satisfy some of these requirements. In addition, IEEEP1003.14 may propose changes and amendments to a variety of relevant standards in order to encourage the specifiers of these standards to add functions and options that accommodate multiprocessing requirements, building on the IEEE P1003.18 {B29} profile.

Standards relevant to the multiprocessing system profile include

- The POSIX pthreads extension (IEEE Std 1003.1c-1995 {117})
- The batch environment standard (IEEE Std 1003.2d-1994 {118})
- The supercomputing checkpoint and restart facilities (IEEE Std 1003.10-1995 {121}) published as extensions to ISO/IEC 9945-1 :1990 {68}

The multiprocessing systems profile will specify both general-purpose computing and multiprocessor-specific standards. General-purpose standards planned or under consideration for the multiprocessing systems profile include

- The IEEE P1003.18 {B29} platform environment profile, including its specifications of ISO/IEC 9945-1 :1990 {68} and ISO/IEC 9945-2 :1993 {69} services
- The Ada language binding (IEEE Std 1003.5-1992 {119}) and FORTRAN-77 language binding (IEEE Std 1003.9-1992 {120}) to ISO/IEC 9945-1 :1990 {68}
- The IEEE Std 1003.1b-1993 {116} realtime extension
- The IEEE Std 1003.1c-1995 {117} POSIX pthreads extension
- The IEEE P1003.1e {B21} POSIX security extension
- The IEEE P1387 system administration standards
- The IEEE Std 1003.10-1995 {121} supercomputing profile
- The IEEE P1003.13 {B27} realtime applications profiles

As other applicable standards emerge, they too will be incorporated in the multiprocessing systems profile. An annex of the profile will address and list relevant emerging standards to provide an idea of the direction of the multiprocessing systems profile.

Multiprocessing-specific requirement areas identified by the IEEE P1003.14 standard developers include

- System administration tools for multiprocessors
- Parallelizing compilers
- Explicit parallelism
- Threads
- Thread-safe libraries
- Message-passing IPC
- Parallel utilities (e.g., `find`, `grep`, `make`, etc.)
- Scheduler controls
- Processor allocation: mandatory and advisory
- Processor binding
- Degree of symmetry: I/O, computation, memory

Standards will be needed for many of these requirement areas. Many of these requirement areas will, therefore, become the subject of an IEEE P1003.14 proposal for a new standardized function or an option in other standards.

7.3 POSIX Interactive Systems AEP

The POSIX Interactive Systems AEP, IEEE P1003.18 {B29}, is based on ISO/IEC 9945-1 :1990 {68} and related standards. It defines the functionality and standards needed for a system that is as similar as possible to the interactive, multiuser development and runtime environment supported by traditional UNIX systems.

The POSIX Interactive Systems AEP is valuable for many users, vendors, programmers, and procurement officers who do not have the time or desire to analyze and specify all the individual interfaces for a system they need. This profile obviates such analysis by enabling the users to point to a single document that specifies exactly what they should order to obtain a system that looks like traditional UNIX systems, except that the POSIX Interactive Systems AEP will be based totally on formal standards. It provides a building block for more complex AEPs, as well as a focal point for harmonization.

The POSIX Interactive Systems AEP consists of

- ISO/IEC 9945-1 :1990 {68} POSIX system API, with a selection of options and definitions of parameters
- ISO/IEC 9945-2 :1993 {69} POSIX shell and utilities, optionally including its User Portability Utilities Option (Section 5)
- At least one of the following language standards: C {67}, Ada {33}, or FORTRAN-77 {103}
- Language bindings to the POSIX services for these selected languages

The POSIX Interactive Systems AEP reflects the goals and intent of the IEEE P1003.18 standard developers by committing to include future specifications when those specifications are completed and approved as standards. Such specifications include system administration, secure/trusted systems extensions, realtime facilities, verification testing facilities, Ada and FORTRAN-77 language bindings, graphical user interfaces, and network interface facilities.

7.4 Supercomputing AEP

The Supercomputing AEP (IEEE Std 1003.10-1995 {121}) is a profile designed to support application and programmer portability in POSIX-based supercomputer environments. The goal of the profile is to allow supercomputer application code to be ported to other sites, reduce the learning curve of users, and encourage production of timely third-party applications.

The need exists for such a profile because of the differences between supercomputing environments and traditional application environments. One difference is that supercomputing jobs are computationally intensive, very long running, and very demanding of resources. Another is that the cost of the supercomputer CPU and many of its peripheral resources is extremely high.

POSIX base standards are not sufficient for supercomputer environments because the historically derived functions in ISO/IEC 9945-1 :1990 {68} cannot adequately manage the use of, and accounting for, a supercomputer or its resources. Furthermore, supercomputers need much better tape handling, multiprocessing, and other capabilities than POSIX or related public specifications initially support.

The Supercomputing AEP identifies POSIX base standards and other relevant standards that support supercomputing requirements. Where none exist, IEEE Std 1003.10-1995 {121} defines the functionality itself, or instigates the formation of a new group to define it. In addition, IEEE Std 1003.10-1995 {121} takes some of the traditional modifications built to allow POSIX systems to run on supercomputers, and it makes those modifications both consistent across supercomputers and portable to users, system administrators, and applications.

Base computing standards required or specified as options by the supercomputing profile (or planned for specification when the standards are completed) include

- The IEEE P1003.18 {B29} platform environment profile, including its specifications of ISO/IEC 9945-1 :1990 {68} and ISO/IEC 9945-2 :1993 {69} services, as well as some languages and bindings
- The IEEE Std 1003.1b-1993 {116} realtime files and asynchronous I/O facility
- The IEEE P1003.1e {B21} and IEEE P1003.2c {B25} POSIX security extensions
- The IEEE Std 1003.2d-1994 {118} POSIX batch environment standard
- The IEEE P1387 system administration standards
- Several graphics standards, including GKS {20}, PHIGS, {56}, CGM {29}, IGES {112}, and PEX {B49} from the X Consortium
- Several programming languages, including standards for C {67}, Ada {33}, or FORTRAN-77 {103}
- The IEEE P1003.1f {B22} Transparent File Access standard for distributed file management

The nonstandardized and nonavailable supercomputing functions identified in the IEEE Std 1003.10-1995 {121} profile include

- Checkpoint recovery
- A resource manager
- A better tape management facility
- Better mass storage/archiving facilities

When IEEE Std 1003.10-1995 {121} was in development, there were no existing standards for batch scheduling and administration facilities. Therefore, IEEE P1003.15 was created to define such facilities.

To meet recovery and archiving requirements, the IEEE P1003.10 standard developers defined system interfaces for functions that perform checkpoint, restart, and better magnetic tape handling (e.g., to rewind a tape under program control). These interfaces have been submitted to IEEE P1003.1 for inclusion in the next ISO/IEC 9945-1 :1990 {68} revision.

7.5 Realtime AEPs

Different types of realtime applications have different characteristics and diverse requirements. For example, embedded systems generally do not need the full functionality of an operating system, nor do they require all the IEEE Std 1003.1b-1993 {116} Realtime Extensions. Compliance with the entire realtime standard or POSIX operating system interfaces or both could reduce the responsiveness of the embedded system and increase the amount of memory required for systems that need to be embedded in limited space. High-end realtime systems, on the other hand, have softer realtime requirements. However, they need the full operating system and realtime functionality.

Therefore, IEEE P1003.13 was created to define profiles for various types of realtime applications. The realtime profiles defined will determine which interfaces can be implemented for a given type of realtime system to claim conformance to the realtime standard.

IEEE P1003.13 is defining profiles to address several types of realtime applications. These are described in the following four subclauses:

- Minimal (Embedded) Realtime System Profile
- Realtime Controller System Profile
- Dedicated Realtime System Profile
- Multipurpose (High-End) Realtime System Profile

7.5.1 Minimal (Embedded) Realtime System Profile

Minimal realtime systems are embedded in stand-alone systems or buried deeply in the overall system electronics, and they are dedicated to the unattended control of one or more special I/O devices. Such systems are typically used for robot controllers, instrumentation, high-speed data acquisition, satellite subsystem control, and flight control. Time-critical responsiveness is a key requirement of embedded systems. Minimal embedded realtime systems, however, have no requirements for a file system, multiple processes, user interaction, or I/O via specific device drivers. Including such features in an embedded realtime system could compromise the ability of the realtime system to satisfy its realtime requirements or to fit into the embedded system physically.

Since embedded systems need only minimal functionality, IEEE P1003.13 identifies a relatively small number of IEEE Std 1003.1b-1993 {116} and ISO/IEC 9945-1 :1990 {68} functions that are required for portable realtime embedded applications. Among other functions, the minimal embedded realtime profile specifies the IEEE Std 1003.1b-1993 {116} semaphores, realtime signals, timers, and synchronized I/O for realtime responsiveness and synchronization; the IEEE Std 1003.1b-1993 {116} message passing interfaces for communication among threads, and the IEEE Std 1003.1c-1995 {117} threads extension to support multiple flows of control. The minimum hardware required is a single processor with its memory. However, no memory management unit (MMU) or common I/O devices are required.

Although subsets of base standards were not allowed in the past, IEEE PASC has made an exception for the IEEE P1003.13 realtime AEP project. Under the new policy, IEEE P1003.13 will be an authorized, special-purpose subset of ISO/IEC 9945-1 :1990 {68} to meet the needs of hard realtime applications. The POSIX subset developers will coordinate their work with IEEE P1003.1.

The ability to subset the ISO/IEC 9945-1 :1990 {68} base standard is particularly important to the POSIX realtime subprofiles that require “small kernels” (e.g., embedded systems profiles, POSIX profiles for controllers, etc.). These

profiles do not need a full-featured ISO/IEC 9945-1 :1990 {68}. In fact, the use of a full-featured ISO/IEC 9945-1 :1990 {68} may cause embedded realtime systems and other very demanding realtime systems to miss their response times.

7.5.2 Realtime Controller System Profile

Realtime controller systems are another example of embedded realtime systems with hard realtime constraints, particularly time-critical responsiveness. They are typically used in control systems (for example, automated systems controllers and process control), as well as some testing.

Realtime controller systems are similar to minimal embedded realtime systems, except that they require a file system and asynchronous (nonblocking) I/O interfaces. Like minimal realtime systems, realtime controller systems require threads, but not multiple processes.

The minimum hardware required for minimal embedded realtime systems is a single processor with its memory, no memory management unit (MMU), and one or more RS-232-like serial channels (for downloading and debugging). Mass storage devices are not required; the file systems may, for instance, be implemented in memory (RAM disk).

7.5.3 Dedicated Realtime System Profile

Dedicated midrange or intermediate-level realtime profiles are targeted at computer-oriented applications that are typically used in avionics, radar systems, submarines, and medical imaging equipment, as well as controllers that manage a group of robots or a subsystem on the factory floor. These applications tend to run on platforms that are dedicated to a single application set or mission mode.

The design of such dedicated realtime applications varies from simple to complex, to accommodate a range of requirements. Such requirements may include sophisticated signal processing capabilities, but they do not necessarily include a file system. A profile that satisfies these requirements would likely specify most of the IEEE Std 1003.1b-1993 {116} and ISO/IEC 9945-1 :1990 {68} functionality (except for hierarchical file system facilities), along with relevant options from the IEEE Std 1003.1b-1993 {116} and ISO/IEC 9945-1 :1990 {68} standards and the IEEE Std 1003.1c-1995 {117} threads extension. Since memory management hardware may be provided, the functionality of memory locking is provided. There also is a common interface for device drivers and files (although there is no file system).

The hardware model for the dedicated (midrange) system profile assumes one or more processors with or without MMUs in the same system.

7.5.4 Multipurpose (High-End) Realtime System Profile

High-end, multipurpose realtime applications are applicable to complex realtime systems. Such multipurpose realtime systems typically are used in military command and control, in space station control systems, in systems that control robot or factory subsystems, as the operating system for high-end simulation systems, and at high-functionality realtime applications that are paced by operator interaction.

The current realtime multipurpose profile is geared to full-function realtime systems, such as simulation applications, and embodies most of the existing practice in the simulator world. Since high-end multipurpose systems have a greater design complexity than embedded or midrange systems, need much greater functionality, and generally support a mix of realtime and nonrealtime processes, the high-end multipurpose realtime profile will require all of IEEE Std 1003.1b-1993 {116} and ISO/IEC 9945-1 :1990 {68}. Since high-end realtime applications may support interactive sessions with users, ISO/IEC 9945-2 :1993 {69} (including the User Portability Utilities Option) is included as a command interface, and the X11 Window system is specified as the basis for an HCI. Support for additional functionality is provided through options for networking and programming languages. IEEE Std 1003.1c-1995 {117} threads support is required so that multitasking may be done by threads, processes, or both.

The hardware model for this profile assumes processors with memory management units, high-speed storage devices, special interfaces, network support, and display devices.

Annex A Bibliography

(Informative)

This annex contains all emerging standards, public specifications, and other references cited in this guide. All formal international, regional, and national standards cited in this guide are listed as normative references in 1.2 and are not repeated here.

{B1}ISO/IEC TR 10000-1: 1992, *Information technology—Framework and taxonomy of International Standardized Profiles—Part 1: Framework*.

{B2}ISO/IEC TR 10000-3: 1994, *Information technology—Framework and taxonomy of International Standardized Profiles—Part 3: Principles and Taxonomy for Open System Environment Profiles*.

{B3}ISO/IEC 11756:1992, *Information technology—Programming languages—MUMPS*.

{B4}ISO/IEC 13719-1:1995, *Information technology—Portable Common Tool Environment (PCTE)—Part 1: Abstract specification*.

{B5}ISO/IEC JTC 1 N1335, *Final Report of ISO/IEC JTC 1 TSG-1 on Standards necessary to define Interfaces for Application Portability (IAP)*, Apr. 1991.

{B6}International Organization for Standardization/Association Française de Normalisation. *Dictionary of Computer Science/Dictionnaire de L'Informatique*. Geneva/Paris: ISO/AFNOR, 1989.

{B7}ANSI/MDC X11.1-1990, *Programming Language MUMPS*.

{B8}CAN/CSA-Z243.200-92²⁴, *Canadian National Keyboard Standard for English and French Languages*.

{B9}DOD5200.28-STD²⁵, *US Department of Defense Trusted Computer System Evaluation Criteria (Stock Number 008-000-00461-7)*.

{B10}EIA/IS-106 (Jan. 1994)²⁶, *CDIF—CASE Data Interchange Format—Overview*.

{B11}EIA/IS-107 (Jan. 1994), *CDIF—Framework for Modeling and Extensibility*.

{B12}EIA/IS-108 (Jan. 1994), *CDIF Transfer Format—General Rules for Syntaxes and Encodings*.

{B13}EIA/IS-109 (Jan. 1994), *CDIF—Transfer Format Syntax—SYNTAX.1*.

{B14}EIA/IS-110 (Jan. 1994), *CDIF—Transfer Format Encoding—ENCODING.1*.

{B15}EIA/IS-111 (Jan. 1994), *CDIF—Integrated CASE Meta-model—Foundation Subject Area*.

{B16}FIPS Publication 120-1,²⁷ *Graphical Kernel System (GKS)*.

²⁴CAN/CSA documents can be obtained from the Canadian Standards Association, 178 Rexdale Boulevard, Rexdale (Tronoto), Ontario, Canada, M9W 1R3. Telephone: +1 (416) 747-4044.

²⁵This document is available from the Superintendent of Documents, P.O. Box 371954, Pittsburgh, PA 15250-7954, USA.

²⁶EIA documents can be obtained from the Electronic Industries Association, 2001 I Street NW, Washington, DC 20036, USA. Telephone: +1 (202) 467-4961

²⁷FIPS publication are available from the US Department of Commerce, National Technical Information Service (NTIS), Springfield, V 22161, USA Telephone: +1 703 487-4650; Fax: +1 703 321-8547.

{B17}FIPS Publication 146-1, *United States Government Open Systems Interconnection Profile (US GOSIP)*.

{B18}FIPS Publication 151-2, *Portable Operating System Interface (POSIX®) System Application Program Interface [C Language]*.

{B19}FIPS Publication 158-1, *User Interface Component of Applications Portability Profile*.

{B20}FIPS Publication 173, *Spatial Data Transfer Standard (SDTS)*.

{B21}IEEE P10031e/D14 Apr. 1994²⁸, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API)—Amendment: Protection, Audit, and Control Interfaces [C Language]*.

NOTE: This draft was formerly called P1003.6.1.

{B22}IEEE P10031f/D6,²⁸ May 1992, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API)—Amendment: Network-Transparent File Access [C Language]*.

NOTE: This draft was formerly called P1003.8.

{B23}IEEE P10031g/D6.3,²⁸ Nov. 1995, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)—Part 1: System Application Program Interface (API)—Amendment: API Network Process-to-Process Communication [C Language]*.

NOTE: This draft was formerly called P1003.12.

{B24}IEEE P10032b/D11²⁸ May 1995, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)—Part 2: Shell and Utilities—Amendment*.

{B25}IEEE P10032c/D14²⁸ Mar 1994 16, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)—Part 2: Shell and Utilities—Amendment: Protection and Control Utilities*.

NOTE: This draft was formerly called P1003.6.2.

{B26}IEEE P10035b/D5²⁸ May 1995, *Draft Standard for Information Technology—POSIX® Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API)—Amendment: Binding for Realtime Extension*.

{B27}IEEE P100313/D6²⁸ June 1994, *Draft Standard for Information Technology—Standardized Application Environment Profile—POSIX® Realtime Application Support*.

{B28}IEEE P100314/D10²⁸ Feb 1995, *Draft Standard for Information Technology—POSIX® Standardized Profile—POSIX Multiprocessor Application Environment Profile*.

{B29}IEEE P100318/D12²⁸ Feb 1995, *Draft Standard for Information Technology—POSIX® Standardized Profile—POSIX Interactive Systems Application Environment Profile*.

{B30}IEEE P12011/D7²⁸ Mar 1993, *Draft Standard for Information Technology—Uniform Application Program Interface—Graphical User Interfaces*.

²⁸This unapproved draft document is available from the IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. Telephone: 1 (800) 678-IEEE or +1(908) 981-1393 (outside USA).

- {B31}IEEE P12012/D2²⁹ Aug 1993, *Draft Recommended Practice for User Interfaces—User Portability/Drivability*.
- {B32}IEEE P13873/D6²⁹ Dec 1994, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®) System Administration—Part 3: User Administration*.
- {B33}IEEE P13874/D8²⁹ Oct 1994, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®) System Administration—Part 4: Print Administration*.
- {B34}IEEE Computer Society Portable Applications Standards Committee. PASC POSIX® *Standards Style Guide*.
- {B35}IEEE Computer Society Portable Applications Standards Committee. *PASC Profile Steering Committee Policies and Guidelines: Rules and Definition Regarding the Development of PASC Standardized Profiles*.
- {B36}RFC-791³⁰ Postel, J., *Internet Protocol*.
- {B37}RFC-793, Postel, J., *Transmission Control Protocol*.
- {B38}RFC-821, Postel, J., *Simple Mail Transfer Protocol*.
- {B39}RFC-822, Crocker, D., *Standard for the format of ARPA Internet text messages*.
- {B40}RFC-854, Postel, J. and Reynolds, J., *Telnet Protocol specification*.
- {B41}RFC-959, Postel, J. and Reynolds, J., *File Transfer Protocol*.
- {B42}RFC-1014, Sun Microsystems, Inc., *XDR: External Data Representation standard*.
- {B43}RFC-1034, Mockapetris, P., *Domain names — concepts and facilities*.
- {B44}RFC-1050, Sun Microsystems, Inc., *RPC: Remote Procedure Call Protocol specification*.
- {B45}RFC-1094, Sun Microsystems, Inc., *NFS: Network File System Protocol specification*.
- {B46}Asente, Paul J. and Swick, Ralph R., *X Window System X Toolkit Specification*, X Version 11, Release 5, Bedford, MA: Digital Press, 1992.
- {B47}Her Majesty's Stationery Office, *Government Open Systems Interconnection Profile (UK GOSIP)*, 1991.
- {B48}ITSEC, *The Information Technology Security Evaluation Criteria*, Version 1.2, 28 June 1991.
- {B49}MIT X Consortium, *PEX Protocol Specification and Encoding Version 5.1P*. Sebastopol, CA: O'Reilly & Associates, 1992.
- {B50}Open Software Foundation. *Application Environment Specification (AES)—Operating System Component (OSC)*. Cambridge, MA: Prentice Hall, 1992.
- {B51}Open Software Foundation. *AES/Distributed Computing: Remote Procedure Call*. Cambridge, MA: Prentice Hall, 1993.

²⁹This unapproved draft document is available from the IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. Telephone:1 (800) 678-IEEE or +1(908) 981-1393 (outside USA).

³⁰Internet Requests for Comments (RFC) are available from the DDN Network Information Center, SRI International, Menlo Park, C 94025, USA

{B52}Scheifler, Robert W. and Gettys, Jim, *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD*, Second Edition. Bedford, MA: Digital Press, 1990.

{B53}University of California at Berkeley—Computer Science Research Group. *4.3 Berkeley Software Distribution, Virtual VAX-11 Version*. Berkeley, CA: The Regents of the University of California, 1986.

{B54}UNIX Systems Laboratory. *System V Interface Definition (SVID), Issues 2 and 3*. Morristown, NJ: UNIX Press, 1986, 1989.

{B55}X/Open CAE Specification C140. *Xlib—C Language Binding*. Reading, UK: X/Open Company, 1991.

{B56}X/Open CAE Specification C150. *X Window System Protocol*. Reading, UK: X/Open Company, 1991.

{B57}X/Open CAE Specification C160. *X Toolkit Intrinsics*. Reading, UK: X/Open Company, 1991.

{B58}X/Open CAE Specification C170. *X Window System File Formats and Application Conventions*. Reading, UK: X/Open Company, 1991.

{B59}X/Open CAE Specification C193. *Distributed TP: The XA Specification*. Reading, UK: X/Open Company, 1992.

{B60}X/Open CAE Specification C201. *Structured Query Language (SQL)*. Reading, UK: X/Open Company, 1992.

{B61}X/Open CAE Specification C206. *Management Protocol Profiles (XMPP)*. Reading, UK: X/Open Company, 1993.

{B62}X/Open CAE Specification C209. *Protocols for X/Open PC Interworking: SMB*. Reading, UK: X/Open Company, 1992.

{B63}X/Open CAE Specification C210. *CPI-C Specification*. Reading, UK: X/Open Company, 1992.

{B64}X/Open CAE Specification C1922. *COBOL Language*. Reading, UK: X/Open Company, 1991.

{B65}X/Open Developer's Specification D030. *Protocols for X/Open PC Interworking: (PC)NFS*. Reading, UK: X/Open Company, 1990.

{B66}X/Open Guide G307, Version 2. *Distributed TP: Reference Model*. Reading, UK: X/Open Company, 1993.

{B67}X/Open Guide G010. *Security Guide (Second Edition)*. Reading, UK: X/Open Company, 1991.

{B68}X/Open Consortium Specification J501. *SQL Remote Database Access Over OSI and Non-OSI Transport Providers*. Reading, UK: X/Open Company, 1995.

{B69}X/Open Publication Set T906. XPG4. Reading, UK: X/Open Company, 1995.

Annex B Standards Organizations and Contact Information

(Informative)

B.1 Introduction

This annex provides a brief summary of the major national and international organizations working on the standardization of information technology.

There are two major categories of open standards organizations. One consists of formally recognized standards bodies, responsible for definition and dissemination of public standards. Their specifications are known as formal or *de jure* standards. International, national, and regional standards groups, and the standards groups of some professional and technical organizations, are examples of formal standards bodies. Organizations specifying standards for open systems usually give precedence to international standards first, then regional, national, and finally professional group standards.

The other standards organization category consists of informal bodies. Informal standards bodies are typically created by suppliers or users of information technology, usually following the consensus method, to enable the implementation of standards. They produce specifications known as industry standards or *de facto* standards. Certain trade associations, industry groups, vendor consortia, and user groups are examples of informal standards bodies. For informal specifications to be approved as formal standards (e.g., international or national standards), informal standards groups typically submit their specifications to formal standards organizations.

The term *de facto standard* is sometimes applied to popular vendor-defined systems. Such systems, however, are closed systems, often controlled in a proprietary fashion. Although they have value, closed *de facto* standards are not the subject of this guide.

Most standards bodies support three types of status for their standards or specifications--approved, draft, and work item. An approved standard is one that has been fully ratified by whatever means the approving standards body uses. A draft standard is one that has yet to be fully ratified, such as a Draft International Standard (DIS) or a CEN ENV (prestandard). Work item is an inclusive phrase for everything else, such as immature specifications and technical reports, that have not yet achieved draft status.

B.1.1 International Standards Bodies Overview

Standards with the highest status are those agreed upon internationally. In information technology, these are produced and published by the Joint Technical Committee 1 (JTC 1), formed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Other standards and/or recommendations are issued by the International Telecommunication Union (ITU) and the ITU-T. Participants in international standards bodies are normally countries, rather than individual suppliers or users, with trade bodies often participating as observers.

B.1.2 National Standards Bodies Overview

Each country has a national body that is the formal representative to the international standards groups. Depending on the country, national bodies receive representatives from trade suppliers or users or both. In general, national bodies support and adopt the international standards, developing national standards only if no international standards are available or to meet special national requirements.

The relationship between the major international and national standards groups is shown in Figure B.1.

B.1.3 International and National Standards Bodies Relationship

Nongovernment standards organizations include trade associations, professional and technical societies, vendor consortia, user groups, and other special interest groups. Actual standards development occurs within these groups. The standards specified by formal standards groups within this category typically are subsequently submitted to national or international standards organizations for approval. Many informal bodies submit their specifications to formal bodies for approval as an accredited standard. (See Figure B.1).

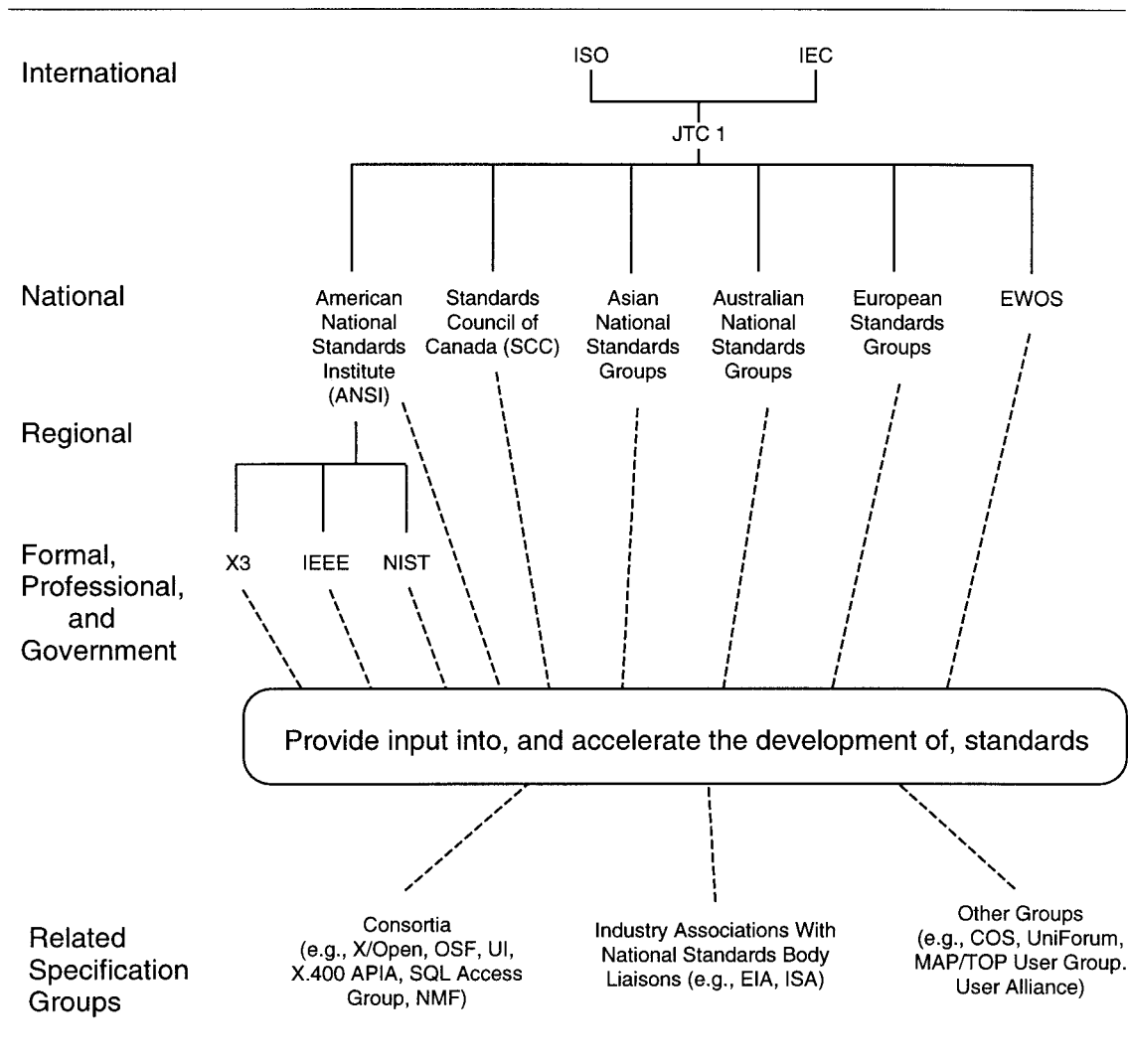


Figure B.1—Selected Major Standards and Standards-Influencing Bodies

B.2 The Formal Standards Groups

B.2.1 International and National Standards Organizations

B.2.1.1 ANSI: American National Standards Institute

ANSI, the national standards coordinating body for the United States, is the standards body that accredits the IEEE (under whose auspices this guide is being developed), as well as accrediting several other standards development groups whose standards are referenced in this guide.

A voluntary organization founded in 1918, ANSI performs three major types of functions. First, ANSI approves standards and accredits standards development groups and certification programs. ANSI does not itself develop standards. Instead, it approves voluntarily submitted specifications that were developed by technical and professional societies, trade associations, and special interest groups, if these specifications and/or groups meet ANSI criteria for due process and consensus.

ANSI accredits three types of organizations:

- Professional societies, such as the IEEE.
- Committees formed for the exclusive purpose of developing standards, such as X3.
- Groups accredited by ANSI to use the canvass method to develop standards. Such organizations prepare a standard using their internal procedures. Then they submit that standard to balloting by other organizations representing a variety of interests. Last, they reconcile comments and objections returned. NIST is an organization accredited to use the canvass process for standards development.

The second major function of ANSI is to represent and coordinate US interests in international, nontreaty, and nongovernmental standards bodies.

The third major function of ANSI is to provide information about national, international, and foreign national standards. ANSI membership is open to manufacturers, organizations, users, and communication carriers. At present, more than 220 professional and technical societies and trade associations that develop standards in the US are ANSI members, as are 1000 companies.

Contact:

American National Standards Institute (ANSI)
11 West 42nd Street
New York, NY 10036
+1 (212) 642-4900
Telex: 42 42 96 ANSI UI
Fax: +1 (212) 302-1246

B.2.1.2 ITU-T: International Telecommunication Union Telecommunication Standardization Bureau

The ITU-T is part of the ITU, which is a United Nations treaty organization. It is now a specialized agency of the United Nations.

The primary mission of the ITU-T is to develop standards supporting the international interconnection and interoperability of telecommunication networks at interfaces with end-user systems, carriers, information and enhanced-service providers, and customer premises equipment. Every four years, the ITU-T publishes the results of its work as "Recommendations." Its recommendations are law where communications in Europe are nationalized.

Membership and participation in the ITU-T are open to private companies; scientific and trade associations; and postal, telephone, and telegraph administrations. The principal participants of ITU-T are telecommunication

administrations and carriers. Scientific and industrial organizations can participate as observers. The US representative is the Department of State.

Contact:

International Telecommunication Union
Telecommunication Standardization Bureau
Place des Nations
CH-1211, Genève 20,
Switzerland/Suisse
Telephone: +41 22 730 5111
Fax: +41 22 733 7256
Telex: 421000 UIT CH

B.2.1.3 ETSI: European Telecommunications Standards Institute

ETSI, founded in 1988, is a voluntary standards organization involved in producing the telecommunications standards necessary to achieve a European unified market. ETSI was established outside the CEN/CENELEC framework. ETSI, however, works with CEN, CENELEC, and the European Broadcasting Union (EBU) in areas of mutual interest.

The voting membership of ETSI consists of postal administrations, along with manufacturers and trade associations, in each of the European Committee for Post and Telecommunications Administration (CEPT) countries. Membership is not restricted to official representatives of member countries. The United States and US companies have been granted observer status.

Standards approved by ETSI are voluntary standards known as ETS (European Telecommunications Standards). ETSI also conducts prestandardization studies, develops technical reports and guidelines, holds conferences, workshops, seminars, and conducts interviews. ETSI interim standards are designated I-ETS.

Contact:

European Telecommunications Standards Institute
B.P. 52
F-06561 Valbonne CEDEX, France
+33 92 94 42 00
Telex: 470 040 F
Fax: +33 93 65 47 16

B.2.1.4 IEC: International Electrotechnical Commission

The IEC is the equivalent of ISO, but for electrotechnical standards. Formed in 1906, the IEC is limited by its charter to issues and standards that relate to electrical and electrotechnical areas. In contrast, ISO is a general organization, which represents all industries.

The IEC is largely concerned with the measurement, testing, use, and safety of electricity. In addition, the organization also has been involved with standards and specifications since its inception, having been formed initially to standardize certain electric power requirements. Although the IEC was created too late to prevent North America and Europe from standardizing on different voltages for routine use (115 and 220 volts), the IEC has gone on to produce a series of widely used standards, ranging from technical standards for electrical design and manufacture, safety features, and comparison metrics to microprocessors, equipment, devices, instrumentation, and media.

The IEC creates its standards in a series of technical committees and subcommittees. There are more than 80 IEC Technical Committees, each of which can have subcommittees and working groups.

As information technology became incorporated in almost every aspect of business and manufacturing, the IEC formed a Technical Committee to handle information technology equipment requirements. As a result, certain IEC work overlapped the work of ISO. The potential for the specification of incompatible standards for the same device or product greatly increased.

To avoid the possibility of incompatible standards, as well as unnecessary work, in 1987 ISO and the IEC agreed to converge many of their information technology efforts to form JTC 1. The IEC now shares its Technical Committee 83 (Information Technology Equipment), as well as its IEC subcommittee 47B (Microprocessor systems) with ISO, through JTC 1.

IEC membership consists of national bodies that represent the interests of users, manufacturers, trade associations, governments, and academic institutions within a country. In contrast to ISO, IEC rules require each member nation to belong to every IEC Technical Subcommittee. Consequently, each member nation has a stake in every IEC standard.

Contact:

International Electrotechnical Commission (IEC)
3, rue de Varembe
CH-1211, Genève 20,
Switzerland/Suisse
+41 22 734 01 50
Fax: +41 22 733 38 43
Telex: 28872 CEIEC CH

B.2.1.5 ISO: International Organization for Standardization

ISO was established in its present form in 1947 with the aim of reaching international agreement on standards. A voluntary standards developing organization, the membership of ISO consists of standards bodies in participating nations. ISO solicits comments from other groups as well, including ECMA and the ITU-T. ISO has a close relationship with the ITU-T, which is, perhaps, the most influential of all the observer groups within ISO.

ISO develops its own standards [e.g., the Open Systems Interconnection (OSI) model and protocols], and also considers items for standardization that were developed in other standards bodies such as ANSI and IEEE (e.g., the IEEE Std 1003.1-1990 API, which became ISO/IEC 9945-1 :1990 {68}).

Contact:

International Organization for Standardization
Central Secretariat
1, rue de Varembe
CH-1211, Genève 20,
Switzerland/Suisse
+41 22 919 0211
Fax: +41 22 919 0300
Telex: 41 22 05 ISO CH

B.2.1.6 JTC 1: Joint Technical Committee 1

JTC 1, established in 1987, is the first joint committee of ISO and IEC. The groups that joined to create this committee were ISO TC97 (Information Processing Systems) and its subcommittees, IEC Technical Committee 83 (Information Technology Equipment), and subcommittee IEC SC47B (Microprocessor systems). The joint committee was formed to eliminate much of the overlap in standardization activities between the two groups and to prevent the creation of incompatible standards for the same device or technology area.

The purpose of JTC 1 is to develop international standards in the areas of information technology systems (including microprocessor systems) and equipment. Microprocessor systems include, but are not limited to, microprocessor assemblies and related hardware and software for controlling the flow of signals at the terminals of microprocessor assemblies.

JTC 1 initially organized its standards work into four major groupings, each of which contains subcommittees that, in turn, contain working groups. Examples of major JTC 1 groupings and subcommittees are

JTC 1 Application Elements Group

- SC1 Vocabulary
- SC7 Software Engineering
- SC14 Data Element Principles
- SC22 Programming Languages, their Environments and System Software Interfaces

JTC 1 Equipment and Media Group

- SC11 Flexible Magnetic Media for Digital Data Interchange
- SC15 Labelling and File Structure
- SC17 Identification Cards and Related Devices
- SC23 Optical Disk Cartridges for Information Interchange
- SC28 Office Equipment

JTC 1 Systems Group

- SC6 Telecommunications and Information Exchange Between Systems
- SC13 Interconnection of Equipment
- SC18 Document Processing and Related Communication
- SC21 Information Retrieval, Transfer and Management for Open Systems Interconnection (OSI)

JTC 1 Systems Support Group

- SC2 Character Sets and Information Coding
- SC24 Computer Graphics and Image Processing
- SC25 Interconnection of Information Technology Equipment (formerly IEC TC83)
- SC26 Microprocessor Equipment (formerly IEC TC47B)
- SC27 Information Technology Security Techniques (grew out of JTC 1 SC20: Data Cryptographic Techniques)

POSIX standardization work is being done within SC22 Working Group 15 (SC22/WG15). A JTC 1 Technical Study Group on Strategic Planning completed a technical study on Application Portability (AP), the TSG-1 Final Report {B5} . The goal of the study was to identify the standards that need to be written or revised to support application portability between hardware and software environments.

JTC 1 is not involved in application-specific information technology areas, such as banking and industrial automation systems, nor is it concerned with microprocessor subsystems covered by the scopes of IEC TC22 on power electronics or TC86 on fiber optics.

JTC 1 has liaison relationships with numerous ISO and IEC Technical Committees, as well as with the ITU-T.

Like ISO, membership in JTC 1 consists of delegations from standards organizations in member countries. More than twenty countries participate in JTC 1 and there also are more than ten observer countries. ANSI holds the secretariat for JTC 1.

Contact:

American National Standards Institute (ANSI)
11 West 42nd Street
New York, NY 10036
+1 (212) 642-4900
Telex: 42 42 96 ANSI UI
Fax: +1 (212) 302-1246

or

International Organization for Standardization
Central Secretariat
1, rue de Varembe
CH-1211, Genève 20,
Switzerland/Suisse
+41 22 919 0211
Fax: +41 22 919 0300
Telex: 41 22 05 ISO CH

B.2.1.7 JTFS: JTC 1 TAG Special Group on Functional Standardization

The JTFS is a subgroup of the US JTC 1 Technical Advisory Group (JTC 1 TAG). JTFS has been assigned US TAG responsibility in the area of functional standardization, specifically serving as the US TAG for the ISO/IEC JTC 1/SGFS.

B.2.1.8 SGFS: Special Group on Functional Standardization

The SGFS is an ISO group under JTC 1 that is responsible for the international standardization process of profiles or functional standards.

Contact: see B.2.1.6.

B.2.2 National Standards Bodies

National standards bodies manage the national standards efforts of their countries, and represent these countries in JTC 1. All countries concerned with standards have a national standards body. Most of these national standards bodies have the same responsibilities:

- The elimination of technical barriers to free trade
- The creation, coordination, approval, and promotion of standards that satisfy the national interests of the country in question
- The accreditation of standards development groups
- The monitoring and coordination with the standards-developing activities of other national organizations
- The performance of certain test and certification functions
- The accreditation of testing and certification organizations
- The handling of selected standards-testing functions from the standards organizations of other countries
- The provision of information about foreign national standards and international standards
- The creation of standards-based regulations
- The representation of the country in question in international standards bodies (e.g., ISO, the IEC, and ITU-T) and, where applicable, in regional standards bodies (e.g., CEN and CENELEC)

It is because the elimination of technical barriers to free trade is such a key objective of the national standards organizations that these standards organizations play an active role in the international standards arena. This helps ensure that the products of a country can be used and accepted internationally.

Most national standards bodies do not themselves develop standards. Instead, they accredit specific standards development groups to specify standards [e.g., X3 and the IEEE accredited by ANSI and the Canadian Standards Association (CSA) accredited by the Standards Council of Canada (SCC)], or they approve voluntarily submitted specifications that were developed by technical and professional societies, trade associations, and special interest groups [e.g., the Electronic Industries Association (EIA) and Instrument Society of America (ISA)].

With but a few differences, membership in the national standards bodies is generally open to manufacturers, businesses, communications carriers, user groups, user organizations (e.g., consumer advocate groups), trade unions, and interested individuals. In some countries, government agencies also are members of the national standards body; in other cases, government agencies have a separate standards organization that is geared to government interests.

Members of the technical committees and working groups developing national standards are drawn from consumers, manufacturers, government agencies, labor, and consultants. Typically, national standards bodies develop, or otherwise handle, standards in such areas as the environment, electrical and electronics, communications and information processing, construction, energy, transportation and distribution, materials technology, production management, consumer safety, and merchandise quality for products produced in the country of the standards body.

In some countries, some standards are mandatory (e.g., wide-area communications standards in many European countries). In other countries, national standards are mandatory for use in the public sector (e.g., certain AFNOR-approved standards in France). In still other countries, the use of standards is voluntary. Strong incentives for accepting standards, however, are provided in some countries because without proof of conformance to the national standards, insurance carriers will not write insurance for a product (e.g., the DIN testing and inspection mark in Germany).

Many countries publish three types of standards: standards that are mandatory for use, especially in the public sector; experimental standards that use new processes or techniques and whose use is voluntary; and informative or guide standards.

The Japanese national standards body, known as the Japanese Industrial Standards Committee (JISC), is a national standards body that is somewhat different from other national standards bodies and has no true counterpart in other nations. One reason for this difference stems from the fact that the JISC has a special relationship with the Japanese government and major manufacturers. For example, the JISC secretariat is the Agency of Industrial Science and Technology, a division of the Ministry of International Trade and Industry (MITI), which plays a central role in Japanese industry. The influence of this centralized national planning structure eliminates many areas of contention, including among companies with multinational branches, and facilitates the ability for Japanese standards groups to gain a consensus. In addition, the involvement of foreign companies in the JISC is limited because of geographic and linguistic differences and because of restrictions on their meaningful participation. Although large-scale manufacturers may participate, user groups and small manufacturers find participation very difficult.

The national standards body participating members of ISO/IEC JTC 1 at the time of publication of this guide are as follows.

NOTE — JTC 1 national body membership and membership contact information will always be in flux and will likely be out of date soon after publication. Current membership and contact information is available from the JTC 1 secretariat (See B.2.1.1 for contact information).

Australia: Standards Australia (SAA)
P.O. Box 1055
Strathfield NSW 2135
Telephone: +61 2 746 4700
Fax: +61 2 746 8450
Telex: 2 65 14 astan aa

Austria: Österreichisches Normungsinstitut
Heinestrasse 38
Postfach 130 A-1021 Wien
Telephone: +43 1 21 30 00
Fax: +43 1 21 30 06 50
Texex: 11 59 60 norm a

Belgium: Institut Belge de Normalisation
Av. de la Brabanconne 29
B-1040 Bruxelles
Telephone: +32 2 734 92 05
Fax: +32 2 733 42 64
Telex: 2 38 77 benor b

Brazil: Associação Brasileira de Normas Técnicas
Av. 13 de Maio n 13-27 andar
Caixa Postal 1680
CEP: 20, 003 - Rio de Janeiro-RJ
Telephone: +55 21 210 31 22
Fax: +55 21 240 82 49
Telex: 213 43 33 abnt br

Canada: Standards Council of Canada
45 O'Connor Street, Suite 1200
Ottawa, Ontario K1P 6N7
Telephone: +1 613 238 32 22
Fax: +1 613 995 45 64
Telex: 053 44 03 stancan ott

China: China State Bureau of Technical Supervision
4, Zhi Chun Road
Haidian District
P.O. Box 8010
Beijing 100088
Telephone: +86 1 203 24 24
Fax: +86 1 203 10 10

Czech Republic:
Czech Office for Standards, Metrology, and Testing (COSMT)
Vaclavske namesti 19
113 47 Praha 1
Telephone: +42 2 24 22 47 34
Fax: +42 2 24 22 47 26

Denmark: Dansk Standard (DS)
Baunegaardsvej 73
DK-2900 Hellerup
Telephone: +45 39 77 01 01
Fax: +45 39 77 02 02

Egypt, Arab Republic of:

Egyptian Organization for Standardization and Quality Control
2 Latin America Street
Garden City, Cairo
Telephone: +20 2 354 97 20
Fax: +20 2 355 78 41

Finland:

Finnish Standards Association SFS
P.O.Box 116
FIN-00231 Helsinki
Telephone: +358 0 149 93 31
Fax: +358 0 146 49 25

France:

Association Française de Normalisation (AFNOR)
Tour Europe
Cedex 7
F-92049 Paris La Défense
Telephone: +331 42 91 55 55
Fax: +331 42 91 56 56
Telex: 61 19 74 afnor f

Germany:

Deutsches Institut für Normung (DIN)
D-10772 Berlin
Telephone: +49 30 26 01-0
Fax: +49 30 26 01 12 31

Hungary: Magyar Szabványügyi Hivatal
Ullői út + 25
H-1450 Budapest 9
Pf.24
Telephone: +361 218 3011
Fax: +361 218 5125

Ireland: National Standards Authority of Ireland
Glasnevin
Ballymun Road
Dublin-9
Telephone: +353 1 837 01 01
Fax: +353 1 836 98 21

India: Bureau of Indian Standards
Manak Bhavan
9 Bahadurshah Zafar Marg
New Delhi 110002
Telephone: +91 11 331 79 91
Fax: +91 11 331 406 2

Italy: Ente Nazionale Italiano di Unificazione
Via Battistotti Sassi 11 / b
I-20133 Milano
Telephone: +39 2 70 02 41
Fax: +39 2 70 10 61 06

Japan: Japanese Industrial Standards Committee
c/o Standards Department
Agency of Industrial Science and Technology
Ministry of International Trade and Industry
1-3-1 Kasumigaseki, Chiyoda-ku
Tokyo 100
Telephone: +81 3 35 01 92 95/6
Fax: +81 3 35 80 14 18

Korea, Republic of:
Bureau of Standards
Industrial Advancement Administration
2, Chungang-dong, Kwachon-city
Kyonggi-do 427-010
Telephone: +82 2 503 79 28
Fax: +82 2 503 79 41
Telex: 2 84 56 fincen k

Mongolia: Mongolian National Institute for Standardization and Metrology
Ulaanbaatar -51
Telephone: +976 1 35 83 49
Fax: +976 1 35 80 32

Netherlands: Nederlands Normalisatie-instituut
Kalfjeslaan 2
P.O. Box 5059
NL-2600 GB Delft
Telephone: +31 15 69 03 90
Fax: +31 15 69 01 90
Telex: 3 81 44 nni ni

Norway: Norges Standardiseringsforbund
 Hegdehaugsveien 31
 Postboks 7020 Homansbyen
 N-0306 Oslo 3
 Telephone: +47 22 46 60 94
 Fax: +47 22 46 44 57

Romania: Institutul Roman de Standardizare (IRS)
 Str. Jean-Louis Calderon Nr. 13
 Cod 70201
 Bucuresti 2
 Telephone: +40 1 211 32 96
 Fax: +40 1 210 08 33

Russia: Committee of the Russian Federation for Standardization,
 Metrology, and Certification (GOST R)
 Leninsky Prospekt 9
 Moskva 117049
 Telephone: +7 095 236 40 44
 Fax: +7 095 237 60 32

Slovenia: Standards and Metrology Institute (SMIS)
 Ministry of Science and Technology
 Kotnikova 6
 SI-61000 Ljubljana
 Telephone: +386 61 131 23 22
 Fax: +386 61 31 48 82

Sweden: ITS, Information Technology Standardization
Electrum 235
S-16440 Kista
Telephone: +46 8 793 9000
Fax: +46 8 751 5653

Switzerland: Swiss Association for Standardization (SNV)
Muhlebachstrasse 54
CH-8008 Zurich
Telephone: +41 1 254 54 54
Fax: +41 1 254 54 74

United Kingdom:
British Standards Institution (BSI)
389 Chiswick High Road
GB-London W4 4AJ
Telephone: +44 181 996 9000
Fax: +44 181 996 7400

United States of America:
American National Standards Institute (ANSI)
11 West 42nd Street
New York, NY 10036
Telephone: +1 212 642-4900
Fax: +1 212 398-0023
Telex: 42 42 96 ansi ui

B.2.3 Other Formal Standards Organizations

B.2.3.1 CEN/CENELEC/CEPT

The Comité Européen de Normalisation (CEN), Comité Européen de Normalisation Electrotechnique (CENELEC), and the European Committee for Post and Telecommunications Administration (CEPT) are European regional standards committees responsible for developing and publishing European standards. CEN is an association of EC

(European Community) and EFTA (European Free Trade Association) members. It is active in making standards of its members into ISO standards and European standards. CENELEC is the counterpart of CEN that deals exclusively with electrotechnical matters. CEPT and ETSI are the CEN counterpart that deal with telecommunication matters.

CEN, CENELEC, and CEPT can be considered the European regional equivalent of ISO for two reasons. First, they have as members the national standards bodies of their eighteen EC and EFTA member states. Second, standards adopted by these organizations have to be implemented in full as national standards, regardless of the way in which the member voted. Any standards that conflict with them have to be withdrawn. CEN members, for example, agree to use its published standards in preference to existing national standards, wherever possible.

CEN, CENELEC, and CEPT were created to improve the competitiveness of European enterprise by removing technical barriers to trade and facilitating the free movement of goods within Europe. To accomplish its aims, CEN, CENELEC, and CEPT perform the following tasks:

- Create and promote European Standards (EN)
- Rapidly create prestandards (ENV) in technology areas in which there is a high level of innovation or where it is felt that future standardization requires basic guidance. ENVs are subjected to an experimental period of up to three years.
- Create harmonization documents (HD) that are more flexible than European Standards so that the technical, historical, or legal circumstances pertaining to each country can be taken into account.
- Set up a framework for European certification that supports the issuing of a European mark of conformity to certain standards and the mutual recognition of test results and inspections.
- Promote the application within Europe of ISO standards and accelerate their production.
- Work in liaison with European professional federations and numerous technical organizations to establish priority standards programs and contribute to the technical work.

Contact:

European Committee for Standardization (CEN)
European Committee for Post and Telecommunications Administration
2 rue Brederode, Suite 5
B-1000 Brussels, Belgium
+322 519 6811
Telex: 26257 CENLEC B

B.2.3.2 ECMA

Established in 1961 to develop data processing standards, ECMA is a trade organization open to any computer firm developing, manufacturing, or selling in Europe. As of 1992, ECMA had 33 members and 14 active Technical Committees.

ECMA contributes to the ISO standards development efforts, in addition to issuing its own standards. ECMA is particularly active in the development of higher layer protocols for OSI networking. It also is developing a standard for a Portable Common Tool Environment (PCTE).

Contact:

ECMA
114 rue du Rhône
CH-1204, Genève
Switzerland/Suisse
+41 22 849 60 00
Fax: +41 22 849 60 01
Telex: 22288

B.2.3.3 EIA: Electronic Industries Association

The EIA is a US trade organization whose membership consists primarily of manufacturers. The EIA has been a standards developer in the areas of electrical and electronic products and components since 1926. Many of its standards have been submitted to ANSI and approved as ANSI standards. The EIA is best known for the RS-232-C standard.

Contact:

Vice President, Engineering
Electronic Industries Association (EIA)
2001 I Street NW
Washington, DC 20006
+1 (202) 457-4900

B.2.3.4 IEEE: Institute of Electrical and Electronics Engineers

The IEEE is a professional scientific, engineering, and educational organization that develops and publishes standards in a variety of computer and engineering areas. The standards published are of three types: full standards, recommended practices, and guides.

- Standards are documents with mandatory requirements.
- Recommended practices are documents of procedures and positions preferred by the IEEE.
- Guides are documents that suggest approaches to good practice, but make no clear-cut recommendations.

The IEEE is accredited by ANSI and can, therefore, submit its standards directly to the ANSI Board of Standards Review for recognition as American National Standards. However, the IEEE has members in more than 150 countries in the world and therefore has significant global participation in its standards program. IEEE also has cross-adoption relationships with other national, regional, and international standards bodies to recognize and use each others' standards when appropriate.

The IEEE Standards Board authorizes, coordinates, and approves all standards projects and oversees cooperative activities with other standards organizations. Standards are proposed and sponsored by technical committees of the IEEE Societies, Standards Coordinating Committees (SCCs), and Accredited Standards Committees, depending on the scope of the work. Management of the actual standards development and balloting is handled by the committees with oversight and assistance from the Standards Department. The individual draft standards are developed in specific working groups reporting to higher level committees, typically (but not exclusively) one working group per standard (see Figure B.2).

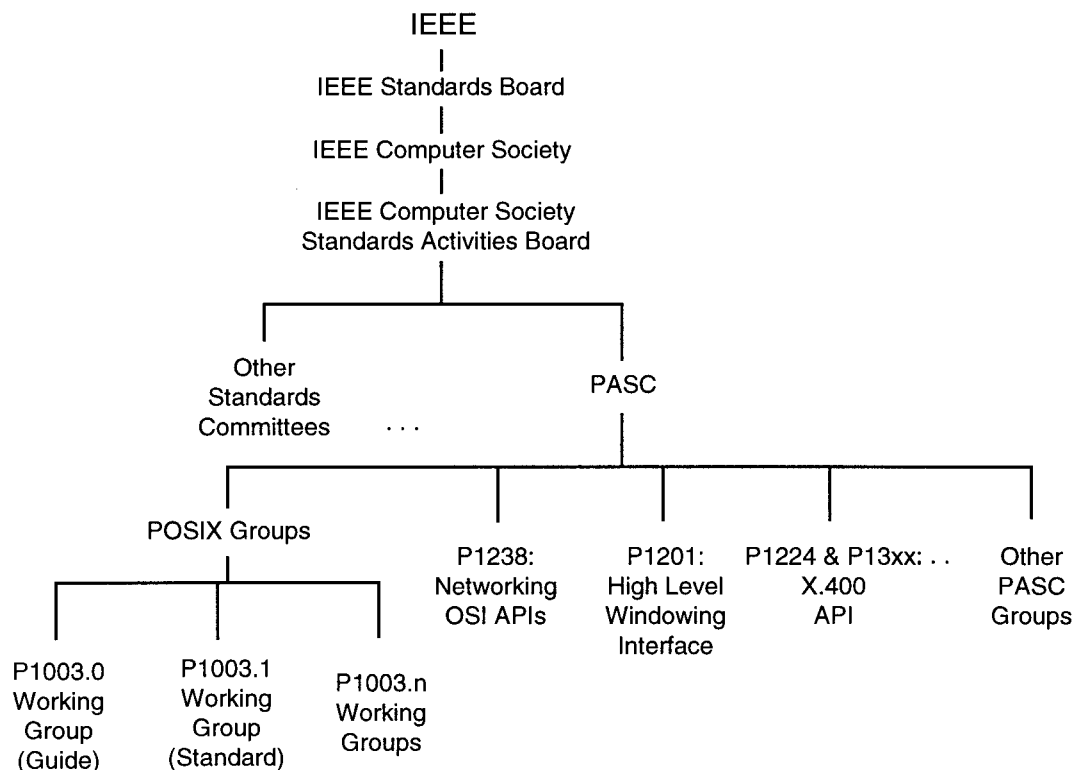


Figure B.2—IEEE Standards Diagram

IEEE membership is open to a broad range of qualified, dues-paying individuals. In most cases, IEEE membership or Computer Society affiliate membership is required for balloting, but not for participating in the development of draft standards. Standards participants are generally individuals, not companies or organizations, except in the accredited Standards Committees that function under the jurisdiction of the Standards Board and in approved Organizational Representation for balloting.

Approximately 30000 individuals are active in IEEE Standards development. More than 600 IEEE standards exist, and more than 800 standards projects are underway. IEEE also administers the Secretariat or Cosecretariat of nine Accredited Standards Committees.

Among the most well-known IEEE Standards are the IEEE 802.3 CSMA/CD and 802.4 token bus LANs, IEEE 488 bus, the National Electrical Safety Code, and the IEEE P1003.n POSIX Standards. Many of the IEEE 802 and 1003 Standards have been approved as ISO/IEC Standards. This document is, in IEEE parlance, a *Guide* to the POSIX OSE.

Contact:

Institute of Electrical and Electronics Engineers, Inc.
 Standards Department
 445 Hoes Lane
 P.O. Box 1331
 Piscataway, NJ 08855
 +1 (908) 562-3800

B.2.3.5 NIST: National Institute of Standards and Technology

The NIST (formerly the National Bureau of Standards) was established by an act of the US Congress on March 3, 1901 to advance, and facilitate the application of, US science and technology for public benefit. Toward this end, the Institute for Computer Sciences and Technology (ICST) within NIST conducts research and provides technical advisory services to help Federal agencies acquire and apply computer technology.

NIST is a major driving force behind standards development. Through the Institute for Computer Sciences and Technology, NIST develops and publishes Federal Information Processing Standards (FIPS) for the United States. Federal agencies are obligated to use these standards, where applicable.

Federal computer standards also are widely used by the private sector and often are adopted as ANSI standards. Besides defining standards, NIST has defined an Application Portability Profile (APP), which comprises a series of nonmandatory specifications and a guide for US government users to apply in developing a portable, interoperable computer architecture and environment.

The development and evolution of both the FIPS program and the APP is carried out in conjunction with users and vendors through an ongoing series of NIST-conducted Implementor Workshops and User Workshops (e.g., OSI implementors workshops, APP workshops, and Integrated Software Engineering Environment workshops). The workshops provide forums for user and vendor feedback and comments on evolving NIST standards, and they help to ensure that there is a general commitment among vendors to building products that conform to the evolving NIST specifications.

Additionally, NIST develops test methods and performance measures to help users and vendors implement standards and to test the conformance of vendor implementations to FIPS specifications. Among others, NIST has test suites for most FIPS programming languages, FIPS Database SQL, and ISO/IEC 9945-1 :1990 {68}. The ISO/IEC 9945-1 :1990 {68}, conformance test suite, however, is based on the conformance-test assertions developed in the POSIX Test and Methods working group (P1003.3.1).

Besides developing its own standards, NIST staff members participate in a number of other standards activities and organizations, including the ANSI X3 Committee on Information Processing Systems, ISO/IEC JTC 1, ITU-T, ECMA, and the IEEE.

Contact:

National Institute of Standards and Technology
Gaithersburg, MD 20899
+1 (301) 975-2000

B.2.3.6 T1

T1, established in 1984, is an ANSI-accredited standards committee that is developing standards and technical reports. The standards and reports are intended to support interconnection and interoperability of telecommunication networks at interfaces with end-user systems, carriers, information and enhanced-service providers, and customer premises equipment.

Six T1 technical subcommittees are currently developing these standards and reports under the T1 Advisory Group. The subcommittees also recommend positions on matters under consideration by other North American and international standards bodies.

T1 Membership and full participation is available to all interested parties.

Contact:

T1 Secretariat
 c/o Exchange Carriers Standards Association
 5430 Grosvenor Lane, Suite 200
 Bethesda, Maryland 20814-2122
 +1 (301) 564-4505
 Fax: +1 (301) 564-4501

B.2.3.7 X3

X3 (Computers and Information Processing Committee), established in 1961, is an ANSI-accredited standards committee that develops computer, information processing, and office systems standards. X3 also participates in the development of international standards in these areas. The Information Technology Industrial Council (ITI) functions as the secretariat for X3.

X3 membership is open to all organizations, upon payment of a service fee. The current membership includes computer manufacturers, communication carriers, user groups, and government agencies. More than 3200 volunteers from these organizations participate in the X3 standards work. They are organized into about 85 technical groups, working on 700 projects.

Three standing committees report to X3: the Long Range Planning Committee (LRPC), the Operational Management Committee (OMC), and the Project Planning Committee (PPC). The following are the major X3 technical committees:

Recognition

X3A1 Optical character recognition

Media

X3B5 Digital magnetic tape
 X3B6 Instrumentation tape
 X3B7 Magnetic disks
 X3B8 Flexible disk cartridges
 X3B9 Paper/Forms layout
 X3B10 Credit/Identification cards
 X3B11 Optical digital data disks

Data management and graphics

X3H2 Database
 X3H3 Computer Graphics
 X3H3.6 Windowing Interfaces
 X3H4 Information Resource & Dictionary
 X3H7 Object Information Management

Languages

X3J1 PL/I
 X3J2 Basic
 X3J3 Fortran
 X3J4 COBOL
 X3J7 APT
 X3J9 Pascal

X3J10 APL
X3J11 C
X3J12 Dibol
X3J13 Common Lisp
X3J14 Forth
X3J15 Databus
X3J16 C++
X3J17 Prolog
X3J18 REXX

Documentation

X3K1 Computer documentation
X3K5 Vocabulary

Data representation

X3L2 Codes and character sets
X3L5 Labels and file structure
X3L8 Data representation

Communication

X3S3 Data communications

Systems technology

X3H5 Parallel processing
X3L3 Audio/Picture coding
X3T3 Open distributed processing
X3T4 Security techniques
X3T6 Noncontact information interfaces
X3T7 Internationalization
X3V1 Office and publishing systems

Contact:

ITIC
c/o X3 Secretariat
1250 Eye Street, NW, Suite 200
Washington, DC 20005-3922
+1 (202) 737-8888
Fax: +1 (202) 638-4922

B.3 Related Organizations

The following organizations are some of the major trade associations, user groups, and professional bodies active in either promoting, implementing, or reviewing information technology standards.

B.3.1 AOW: Asia-Oceanic Workshop

The AOW was chartered in 1988 with the goal of developing ISPs. The secretariat for AOW is INTAP (see B.3.6). Members of AOW come from the Pacific rim. Plenary meetings are held twice per year, usually in Japan. The AOW

includes ten Special Interest Groups (SIGs). SIG meetings may be held in other countries. AOW is the focal point in the Pacific rim for the study and development of operating system profiles. AOW also is a participant in the regional workshop Coordination Committee (AWS-CC), which works to harmonize ISPs across different regions.

Contact:

Asia-Oceania Workshop (AOW)
c/o INTAP
Attn Mr. S. Kurokawa
Sumitomo Gaien Bldg 3F
24 Daikyo-cho, Shinjuku-Ku
Tokyo 160
Japan
Telephone: +81 3 3358 2721 x300
Fax: +81 3 3358 4753

B.3.2 CODASYL: The Conference on Data Systems Languages

CODASYL has been active since 1960 in the development of the COBOL language through its COBOL Committee (CC). Since 1969, it also has been active in the development of a common Data Description Language for defining schemas and subschemas, and in a data manipulation language through the DBTG Data Base Task Group of the CC. The activities of the CC are documented in the COBOL Journal of Development, which serves as the official COBOL language specification.

In 1969, ANSI (then the United States of America Standards Institute) issued the first COBOL standard. At that time, the X3J4 committee stated that X3J4 recognized the CODASYL COBOL Committee as the development and maintenance authority for COBOL. In practice, this meant that ANSI agreed not to make any changes to the CODASYL-defined language specification. Although this agreement has been challenged over the years, the CODASYL-ANSI agreement is still strong. As a result, the CODASYL has enormous influence upon the COBOL language.

Toward the end of 1971, a new CODASYL committee was established—the Data Description Language Committee (DDLC). The DDLC was formed to serve the same functions for the schema DDL as the CC does for COBOL. That is, since the schema DDL is a conceptual schema and network-model database language for use with many programming languages, not just COBOL, the DDLC continues the schema DDL development; and publishes its own Journal of Development documenting the current status of the language.

The COBOL DML and subschema DDL (for defining an external view) of the DBTG are COBOL-specific and have remained part of the CC under the name “The COBOL Data Base Facility.”

The CODASYL membership is composed of voluntary representatives, mostly from computer manufacturers and users in industry and the US Federal government.

Contact:

Dr. Daniel R. Frantz
CODASYL Committee
DEC ZKO2-2/O23
110 Spit Brook Road Nashua,
NH 03062-2698
+1 (603) 881-2272

B.3.3 EPRI: Electric Power Research Institute

EPRI is a research institute supported by electric power utility companies. Its membership comprises more than 673 publicly and privately owned utilities in the United States. Besides providing a variety of utility-specific services to its membership, the latest mission of EPRI is to facilitate the use of open systems technology in the utility industry.

Toward this end, EPRI has developed a Utilities Communication Architecture (UCA), which is similar to the NIST Government Open Systems Interconnect Profile (GOSIP) Version 2.0. Much of the UCA was developed by EPRI with the cooperation of Honeywell and Anderson Consulting.

The specific open system interests of EPRI span realtime UNIX systems, expert systems, and database access using RDA and SQL. Because of the financial structure of the utilities industry, EPRI wants to encourage these and other open systems technologies for equipment with a 12-15 year life cycle.

Contact:

Electric Power Research Institute
3412 Hillview Avenue
P.O. Box 10412
Palo Alto, CA 94304
+1 (415) 855-2000

B.3.4 ESPRIT: European Strategic Programme for Research and Development in Information Technology

ESPRIT is a European research program initiative, started in 1982 and sponsored by the Commission of the European Communities. About 227 projects were implemented during the first phase of the project in five major work areas: advanced microelectronics, software engineering and technology, advanced information processing, office automation, and computer integrated manufacturing.

Nearly thirty projects have enabled substantial advances to be made in establishing internationally recognized standards. Examples are the Portable Communications Tool Environment (PCTE) project, the Communication Network for Manufacturing Applications (CNMA) project, and the Herode project, which has prepared an Office Document Architecture standard for adoption as an ISO standard.

The second phase of the ESPRIT program will be concerned mainly with three areas—microelectronics and peripheral technologies; the creation of technologies and tools for the design of information processing systems; and enhancing the capacity for using and integrating information technology to extend the scope of its applications.

Contact:

ESPRIT
Director General
DG XIII, CEC
rue de la Loi 200
B-1049 Brussels, Belgium
+32 2 235 11 11
Telex: 21877 comeu b

B.3.5 EWOS: European Workshop on Open Systems

EWOS is an ongoing regional workshop, formed in 1987, to provide and coordinate European input to the international standard profiles process. It was formed as the result of an initiative of SPAG, in conjunction with CEN/CENELEC.

EWOS is the focal point in Europe for the study and development of OSI profiles and corresponding conformance test specifications. EWOS documents have only to be submitted to public enquiry by CEN and CENELEC before becoming European norms.

Contact:

European Workshop on Open Systems (EWOS)
rue de Stassart, 36
B-1050 Brussels, Belgium
+32 2 511 74 55
Fax: +32 2 511 87 23

B.3.6 INTAP: Interoperability Technology Association for Information Processing

INTAP is a Japanese national agency, funded by MITI. It deals with information technology, and specifically OSI products and advanced projects. INTAP is developing and providing conformance testing tools and services in Japan in cooperation with POSI.

Contact:

INTAP
Sumitomo Gaien Bldg., 3F
24 Daikyo-cho, Shinjuku-ku
Tokyo 160, Japan
+81 33 358 2721
Fax: +81 33 358 4753

B.3.7 Internet Society

The Internet Society is an organization that provides the focus for much of the research and development underlying all of the suite of Internet protocols and applications.

The Internet is the global public domain network that primarily runs the suite of TCP/IP protocols and applications [known as the Internet Protocol Suite (IPS)]. As proven implementations of OSI progress, the IPS is progressively recognizing OSI standards.

The Internet community comprises the users of the Internet. The Internet community produces documents for communicating details of protocols: Management Information Bases (MIBs), other information needed for system interoperability, and, just recently, portability. There are four main types of Internet documents:

- RFCs (Request for Comment) (de facto standards)
- Informational RFCs
- Experimental RFCs
- Internet drafts

The Internet Standards are de facto standards based on proven and interoperable implementations.

The Internet Society was recently formed to be the umbrella organization for the already existing Internet Architecture Board (IAB) and Internet Engineering Task Force (IETF).

Until the end of 1992, the IAB of the Internet Society was the public domain network standardization body. It performed a role similar to that performed by the ISO Council, or ITU-T Plenary. It now performs a more advisory and strategic architectural role, while the Internet Engineering Steering Group (IESG) (described later) is responsible for authorizing Internet standards or RFCs.

The Internet Engineering Task Force (IETF) is the protocol engineering, development, and standardization arm of the Internet Community. Each IETF Working Group (WG) conducts most of its business via e-mail and normally meets at the plenary meetings that the IETF holds three times a year. The goal of the IETF could be summed up as producing, reviewing, and documenting technology (in that order). Its methods differ from those of ISO. The IETF documents, called Internet RFCs, are available electronically or in hard copy from the Network Information Systems Center at SRI International (Menlo Park, CA). These RFCs are the major source of accurate information about TCP/IP protocols and TCP/IP network applications.

The Internet Engineering Steering Group (IESG) comprises the area director for each of the nine technical areas into which the WGs are organized, together with the IETF Chair and the director for standards.

B.3.8 ITIC: Information Technology Industry Council

ITIC is a trade organization whose primary function is to represent large manufacturers of hardware-based information technologies equipment in lobbying about public policy. In addition, it provides education programs, information exchange forums, and deals with the public image of the industry.

ITIC has long had an interest in standards. It serves as the secretariat for X3. It also offers a standards and technology program where its members can exchange information on standards issues and industry standards.

The members of ITIC are mostly large manufacturers. Members are either American companies or US subsidiaries of non-American companies.

Contact:

ITIC
1250 Eye Street, NW, Suite 200
Washington, DC 20005-3922
+1 (202) 737-8888
Fax: +1 (202) 638-4922

B.3.9 MAP/TOP User Group: Manufacturing Automation Protocol and Technical Office Protocol

The MAP Task Force was formed in 1980 to identify a common OSI-based networking standard for plant-floor systems. Its specifications are known as Manufacturing Automation Protocol (MAP).

The MAP specifications mostly reference OSI standards, but they also draw on ANSI, IEEE, EIA, ITU-T, and various industry standards. Where standards do not exist, as in the case of the manufacturing messaging protocol, the MAP Task Force is either defining its own or instigating their formation by industry standards bodies.

In 1984, the MAP Users Group was formed, under the auspices of GM, with the Society of Manufacturing Engineers as its Secretariat. Its objective is to promote knowledge and use of MAP, and to insure input from users.

In 1985, Boeing sponsored a similar effort to specify common networking protocols, known as the Technical Office Protocols (TOP), for engineering and business offices. TOP is largely compatible with MAP, differing only at the lower two layers and the application layer, where TOP addresses requirements of the technical and office user rather than factory users.

Later in 1985, a TOP Users Group was formed. The entire effort became an international effort known as MAP/TOP, and the user group became the MAP/TOP User Group, which meets twice a year.

Today, the MAP/TOP User Group is an independent, self-funded organization that represents thousands of users worldwide, tied together through a worldwide federation of MAP/TOP user groups. Membership is open to either users or companies. The Industry Cooperative Services (ICS) is the worldwide secretariat.

The MAP/TOP Task Force and User Group also is a major contributor of technical and conceptual ideas and specifications to NIST and many of the IEEE POSIX Groups.

Contact:

World Federation of MAP/TOP Users Groups
P.O. Box 1457
Ann Arbor, MI 48106
+1 (313) 769-4571
Fax: +1 (313) 769-4064

B.3.10 NMF: Network Management Forum

A vendor-driven group, the NMF is chartered to produce a commonly agreed-upon specification subset of ISO network management protocols and the command sets to implement this subset. The promise of the NMF is that all of the network management products that its members come up with will conform to this common specification.

The NMF itself will produce no products nor will it specify implementations. Rather, the NMF will specify interfaces.

Major vendors belong to the NMF from both the computer and telecommunication industries. The NMF has published Release 1 of its specifications (1990). Member firms are developing products that conform to Release 1.

Contact:

Network Management Forum
40 Morristown Road
Bernardsville, NJ 07924
+1 (201) 766-1544
Fax: +1 (201) 766-5741

B.3.11 OMG: Object Management Group

Founded in 1989 and headquartered in Framingham, MA, with marketing operations in Boulder, CO, the OMG is an international organization of more than 146 systems vendors, software developers and users. The OMG was founded to promote the theory and practice of object management technology in the development of software.

The goal of OMG is to develop a common framework, based on industry-derived guidelines, that is suitable for object-oriented applications. The adoption of this framework will make it possible to develop a heterogeneous applications environment across all major hardware and operating systems.

The OMG members are quick to form a consensus on certain object management issues because they see these issues directly affecting their software sales. For example, the OMG object request broker design--key software needed to allow disparate open systems to request object services from remote sites--is supported by most major object-oriented software vendors.

Contact:

Object Management Group
492 Old Connecticut Path
Framingham, MA 01701
+1 (508) 820-4300
Fax: +1 (508) 820-4303

B.3.12 OSF: Open Software Foundation

OSF is a not-for-profit, international research and development organization with the objective of producing technology that enables equipment from various vendors to work together and information and applications to be accessed and shared across a distributed, heterogeneous computing environment. Its goals also include the development of software specifications and test suites for an open computing environment.

OSF specifications are defined, and software developed, using an open process into which vendors and users have input and access. The resulting AES specifications are available in the public domain, and the software licensable, to OSF members and nonmembers under identical terms. Both members and nonmembers can also submit technologies to the OSF for consideration as an OSF specification and/or offering. OSF specifications and software will be based on the POSIX system API standard (ISO/IEC 9945-1 :1990 {68}); a variety of international, national, and industry standards; and other consortia specifications. The remainder of OSF software and specifications will be based on technologies contributed by numerous companies and universities as part of the OSF open process.

OSF active-participation membership is open to user organizations, computer hardware and software suppliers, government agencies, educational institutions, and other interested organizations worldwide.

Contact:

Open Software Foundation
Eleven Cambridge Center
Cambridge, MA 02142
+1 (617) 621-8700

or

Open Software Foundation/Europe
Stefan-George-Ring 29
8000 Munich 81, Germany
+49 89 930 920

or

Open Software Foundation/Japan
ABS Building
2-4-16 Kudan
Minami, Chiyoda-Ku, Tokyo 102, Japan
+81 3 3 221 9770

B.3.13 OIW: Open System Environment Implementors Workshop

The OIW is an expansion of the OSI Implementors Workshop, also known as OIW. Because the requirements of distributed, multivendor systems go well beyond OSI networking, the OIW has expanded its scope and revised its charter to encompass the entire open system environment.

The new charter of the OIW adds the means to address the common applications development environment on multiple vendor systems, in addition to OSI-based interoperability among multiple vendor systems. To achieve its goals, the OIW has established an OSE technical committee to address the following open systems requirements:

- Consider a framework for an OSE
- Provide a series of workshops to generate interest in open system environment specifications
- Propose the use of public domain specifications

The workshops are patterned after the OSI Implementors Workshops, which clarified and, if necessary, defined all the parameters necessary for realworld implementations of OSI products. The new Open System Implementors workshops do likewise for an OSE.

The expanded OIW relies on outside mechanisms to collect, synthesize, prioritize, and deliver OSE user requirements to the OIW. For its part, the OIW focuses on implementation requirements for international (and other formal standards), as well as the identification of gaps in the international standards and the identification of what it calls “public domain specifications” to fill those gaps.

Public domain specifications are specifications that are available and have achieved consensus outside of the formal standards-making bodies. Unlike “fully” proprietary specifications, however, public domain specifications exercise no restraint on who can use the specifications and how they can be used.

Proposed public domain specifications can neither overlap with, nor conflict with, any existing formal standard or formal standards under development. Also, if a public domain specification adds functionality, then it has to be engineered to augment the formal standards in such a way that interoperability among systems implementing the formal standards is not precluded. Furthermore, if a public domain specification is approved by the OIW for use in an OIW Implementors Agreement, the referenced version of the specification may not be modified except as required to fix technical and editorial errors. An enhanced version of a public domain specification is treated as a completely new public domain specification.

Even with these restrictions, referencing such specifications in the OIW Implementors Agreements will be an exceptional, rather than a routine, event. Most OIW specifications and Implementor Agreements will reference formal standards.

The OIW is open to all individuals and expert groups from anywhere who wish to contribute to the open systems efforts. Should technical work overlap with other regional workshops (e.g., EWOS), the OIW will collaborate with these other workshops, just as it did in the past. The OIW will continue to be cosponsored by NIST and the IEEE Computer Society, with the NIST acting as the secretariat.

Contact:

OIW Secretariat
c/o NIST
Attn Mr. Hungate
Bldg 225, Room B226
Gaithersburg, MD 20899
Telephone: +1 301 975 2245
Fax: +1 301 926 3696

B.3.14 POSC: Petrotechnical Open Software Corporation

POSC was founded in 1990 by a group of major oil companies to facilitate the development of integrated computing technology for the exploration and production (E & P) segment of the international petroleum industry. Today, membership is open to all entities interested in the E & P industry. These members include other petroleum companies, E & P service companies, software vendors, computer manufacturers, and research institutes.

The primary mission of POSC is the development of an industry-standard, open systems-based, software integration profile for E & P applications. This platform will be the interface between petrochemical software applications, database management systems, workstations, and users. POSC activities focus on the development of an integrated E & P data model, a common look and feel user front-end, and a set of test suites enabling developers to evaluate their offerings against selected industry standards.

POSC is moving quickly and has sent out two public requests for inputs in several technical areas. Project teams for base standards, the E & P data model, and data access are in place. Staffing is in progress for other projects and special interest groups have been formed. POSC offerings will be released to industry for production over the next few years.

Contact:

Petrotechnical Open Software Corporation
10777 Westheimer, Suite 275
Houston, Texas 77042
+1 (713) 784-1880
Fax: +1 (713) 784-9219

B.3.15 SPAG: Standards Promotion and Application Group

SPAG, founded in 1983, is a nonprofit, international research and development consortium of about 65 information technology manufacturers and users. In 1986, it became a company registered under Belgian law as SPAG Services s.a. The goals of SPAG are to promote multivendor, interoperable products based on international standards, particularly OSI, and to keep its members informed about the latest developments in functional standards and conformance testing of products.

To achieve its goals, SPAG plays a leading role in the European Workshop on Open Systems (EWOS), publishes the Guide to the Use of Standards (GUS) regularly, and participates in the development of ISPs. SPAG is particularly active in the development and marketing of test tools for manufacturing applications. Through its SPAG-CCT efforts, (a collaboration of European organizations) which arose out of the ESPRIT Project 955, SPAG is developing, and will be providing, conformance test tools for testing MAP/TOP 3.0, and conformance testing services to industry.

SPAG also is working within Europe to implement the certification infrastructure for OSI products and is involved in a number of Conformance Test Services (CTS) projects within the Commission of European Communities (CEC). In addition, SPAG is active in Telecommunications areas and is leading a consortium developing verification services for the Broadband Networks project RACE.

SPAG has close working relationships with POSI, its counterpart in the Far East.

Contact:

Standards Promotion and Application Group (SPAG)
Avenue des Arts
1-2 bte 11
1040 Brussels, Belgium
+32 2 210 08 11
Fax: +32 2 210 08 00

B.3.16 SQL Access Group

The SQL Access Group is a vendor group formed by a number of people in the ISO Remote Data Access (RDA) Group.

The charter of the SQL Access Group is broad. First, the Group is chartered to define a common subset of SQL functions to reconcile the many SQLs that exist. The specifications will include an SQL data format, as well as protocols for moving data within a multivendor SQL environment. Also included will be specifications for an enhanced SQL programming interface that will let developers write a single application that can access a variety of SQL databases.

The second charter of the SQL Access Group is to accelerate the work of the RDA group. Third, the SQL Access Group is working on putting more distributed functionality into RDA. Toward this end, each thing accomplished by the SQL Access group is fed back into the RDA group.

Contact:

SQL Access Group
4699 Old Ironsides Drive, Suite 450
Santa Clara, CA 95054
+1 (408) 988-3545
Fax: +1 (408) 988-6712

B.3.17 UniForum

UniForum is a nonprofit international association of open systems professionals. Founded in 1980 as/usr/group, the association has, through its standards committees and technical committees, provided contributions to various standards and continues to be involved in the formal standards development process. The specifications and standards to which UniForum has contributed include the following:

- The 1984/usr/group Standard was contributed as a base document for the IEEE P1003.1 work.
- The UniForum Technical Committee on Real Time meets with IEEE P1003.4, working on the emerging POSIX realtime standards.
- The UniForum Technical Committee on Supercomputing evolved into IEEE P1003.10 .
- The UniForum Technical Committee on Internationalization has contributed specifications to IEEE P1003.1 and IEEE P1003.2 and the X3J11 standard C committee and continues to be a technical resource for both formal and informal standards development bodies.

Contact:

UniForum
2901 Tasman Drive
Santa Clara, CA 95054
(800) 255-5620 or
+1 (408) 986-8840
Fax: +1 (408) 986-1645

B.3.18 X/Open

X/Open is an independent, nonprofit consortium formed in 1984. Its goals are to determine user and market requirements and to specify a complete, source-level-portable application environment and test suites. Although its members were initially vendors, the membership of X/Open now encompasses users, system integrators, value-added resellers, government agencies worldwide, other industry-standards groups, and academic and research institutions.

The X/Open environment includes specifications for an operating system interface, networking, data management, programming languages, floppy disk formats, internationalization, and distributed transaction processing. The X/Open Group does not normally define specifications for these areas. Instead, it chooses from existing and emerging standards. An X/Open market research program and open user requirements congress identify and prioritize user and market requirements, based on input solicited from users. These prioritized requirements are published in a document known as the *Open Systems Directive*. These prioritized requirements also help drive the X/Open specification process.

The X/Open environment is based on the POSIX system API standard (ISO/IEC 9945-1 :1990 {68}), parts of the UNIX Systems Laboratory System V Interface Definition (SVID {B50}), and formal international standards that are already accepted or likely to be accepted. However, to get standards into the field rapidly for practical use, where no formal standards exist, X/Open publishes specifications that are based on widely accepted de facto standards. The

formal standards adopted by X/Open, together with the X/Open specifications, form the Common Application Environment (CAE). Periodically, these are grouped together to form versions of the X/Open Portability Guide (XPG).

Contact:

X/Open Company Ltd.
Apex Plaza
Forbury Road
Reading, Berkshire RG1 1AX
United Kingdom
+44 734 508 311

or

X/Open
1010 El Camino Real
Menlo Park, CA 94025
+1 (415) 323-7992