

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.n	System Administration (formerly P1003.7.n)	5.3
P2003.n	Test Methods (formerly P1003.3.n)	

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 PIIIs	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 Varembé, 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.

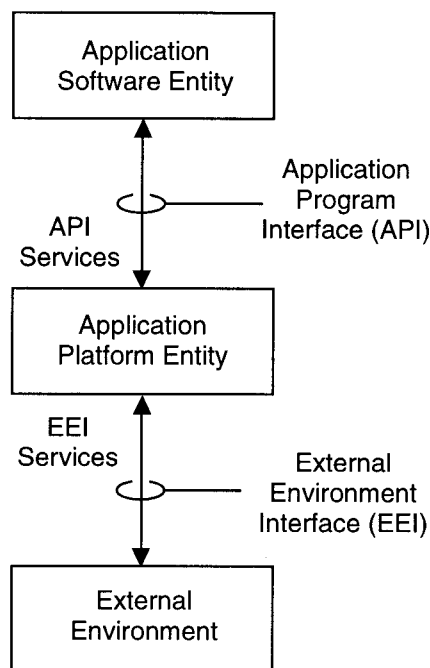


Figure 3.1—POSIX OSE Reference Model

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

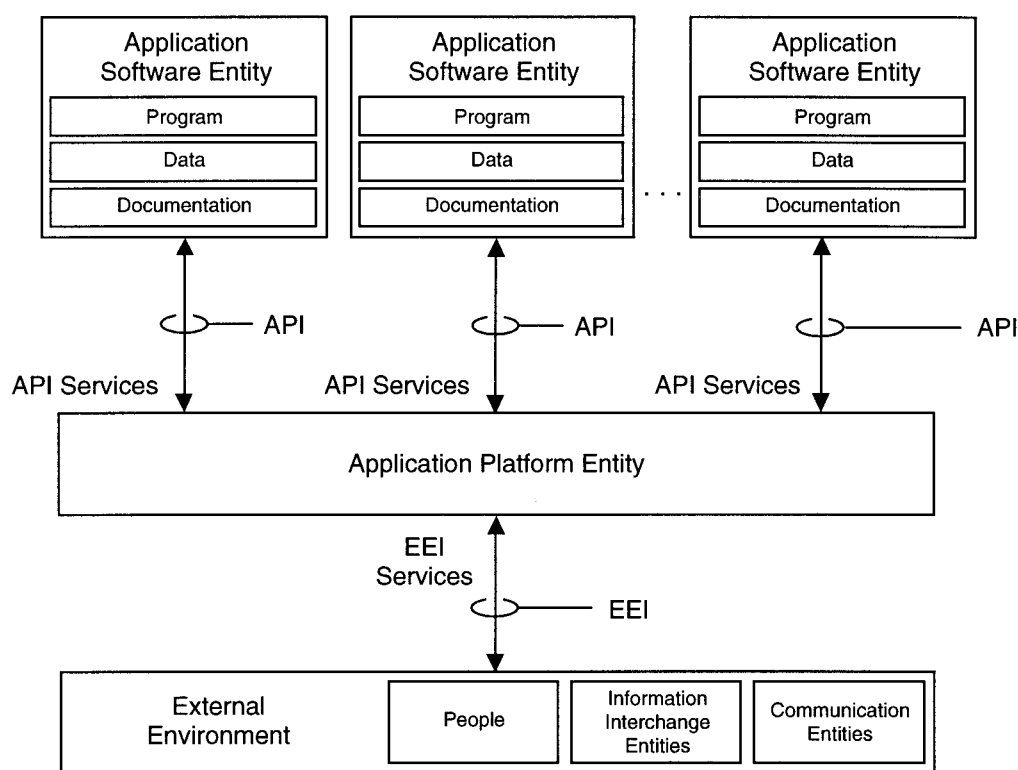


Figure 3.2—POSIX OSE Reference Model — Entities

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.

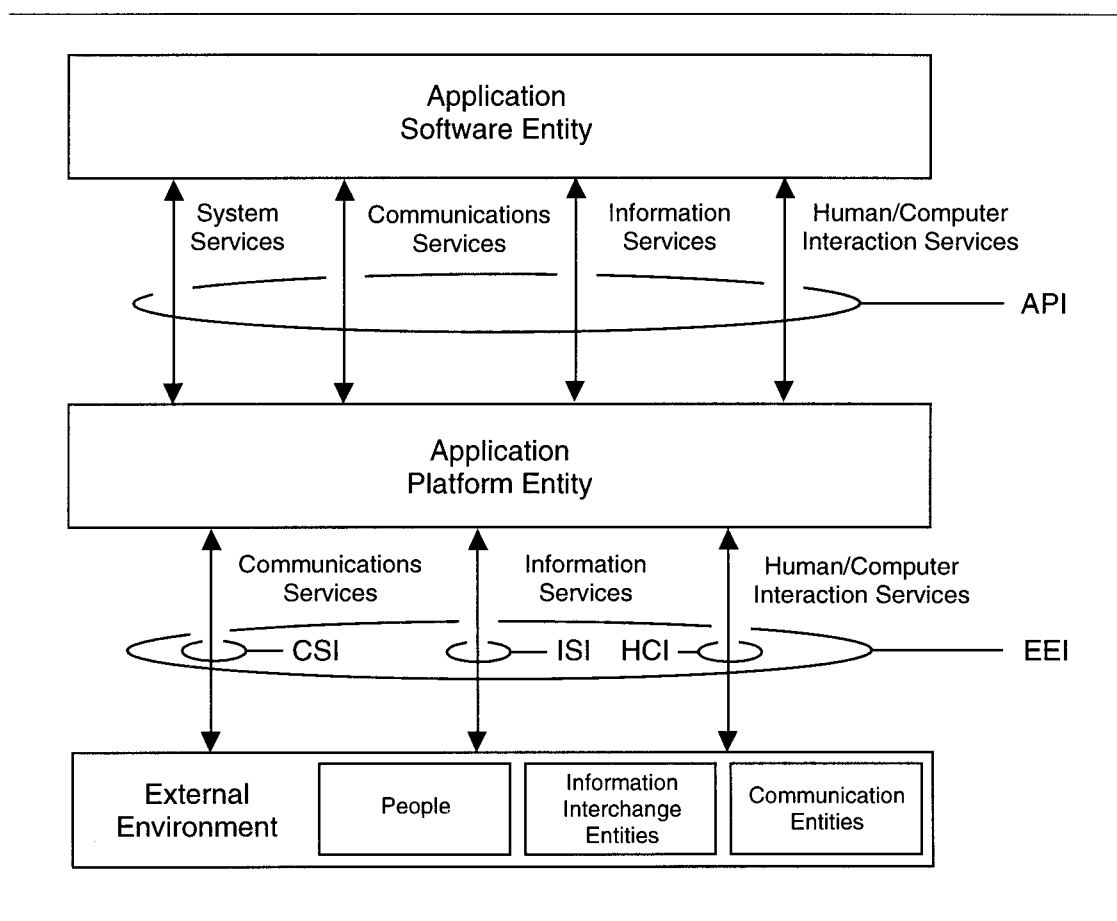


Figure 3.3—POSIX OSE Reference Model — Interfaces

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.

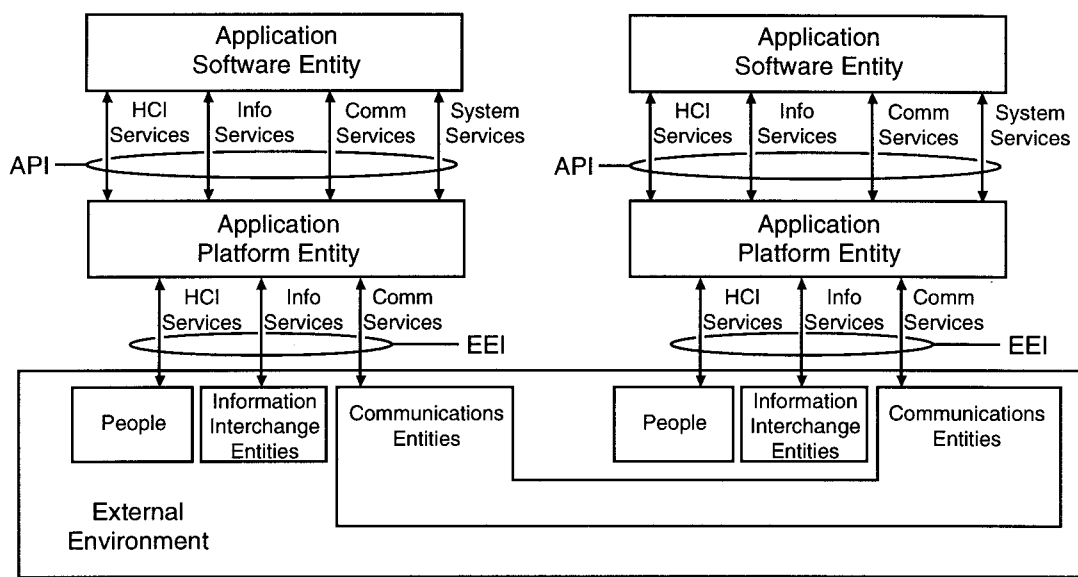


Figure 3.4—POSIX OSE Reference Model — Distributed Systems

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.

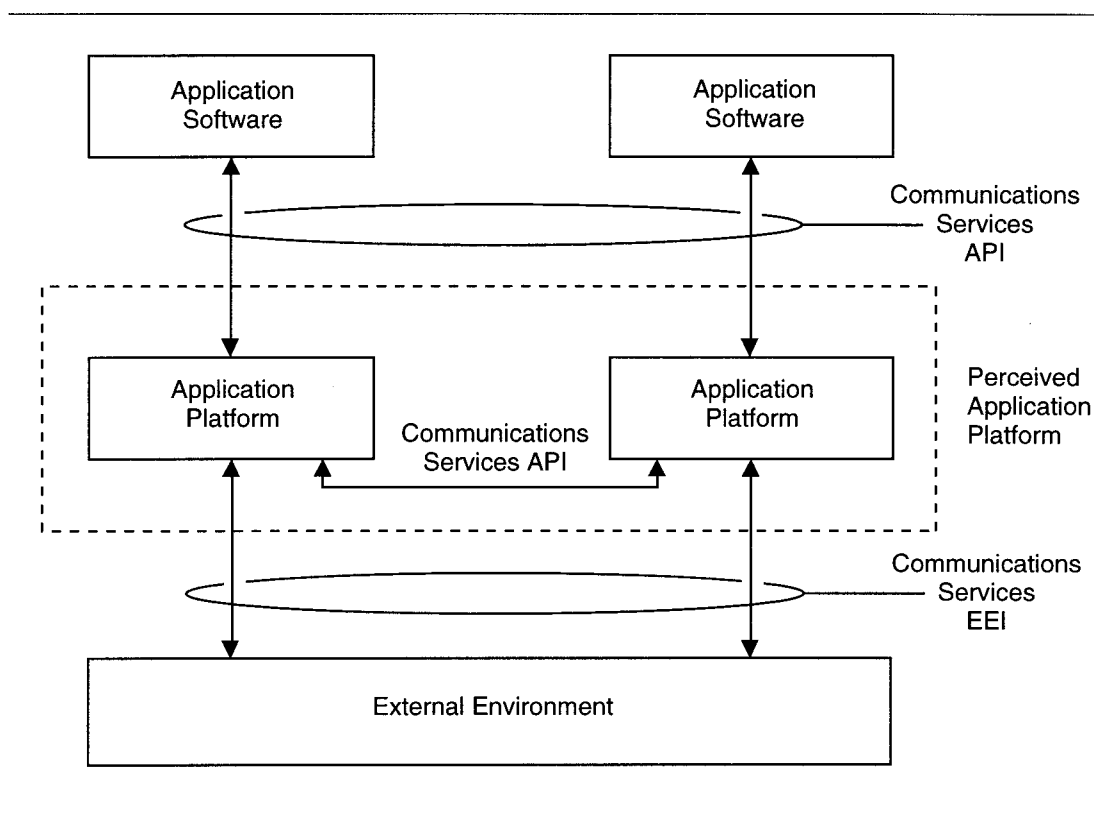


Figure 3.5—Distributed Application Platform Implementation

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 PIIs. Figure 3.6 shows how a PII can be represented in the model.

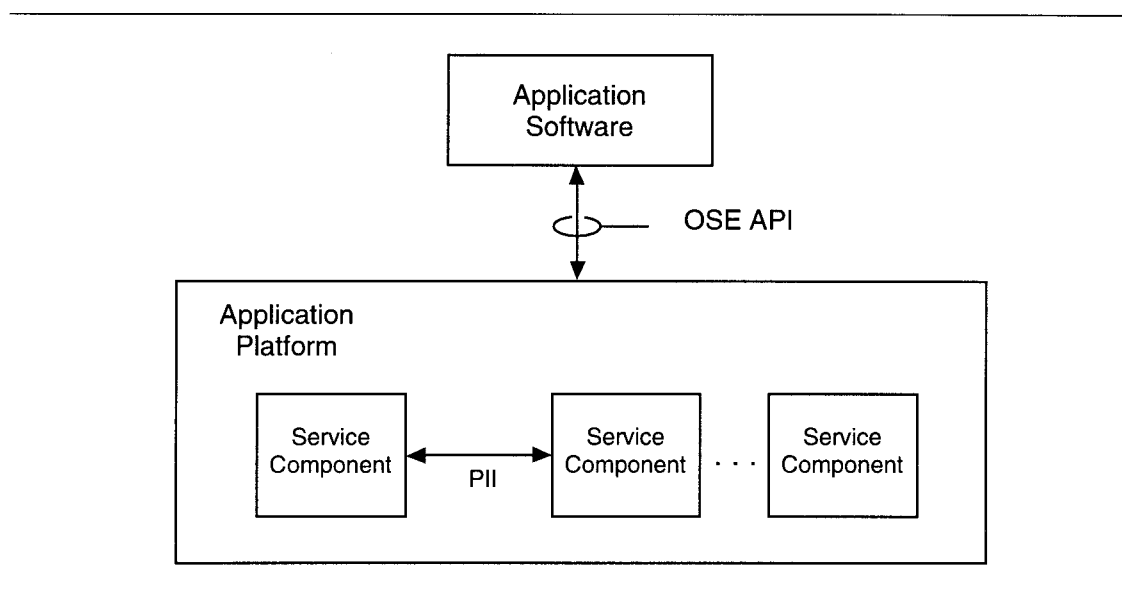


Figure 3.6—Service Components and Interfaces

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 PIIs 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.

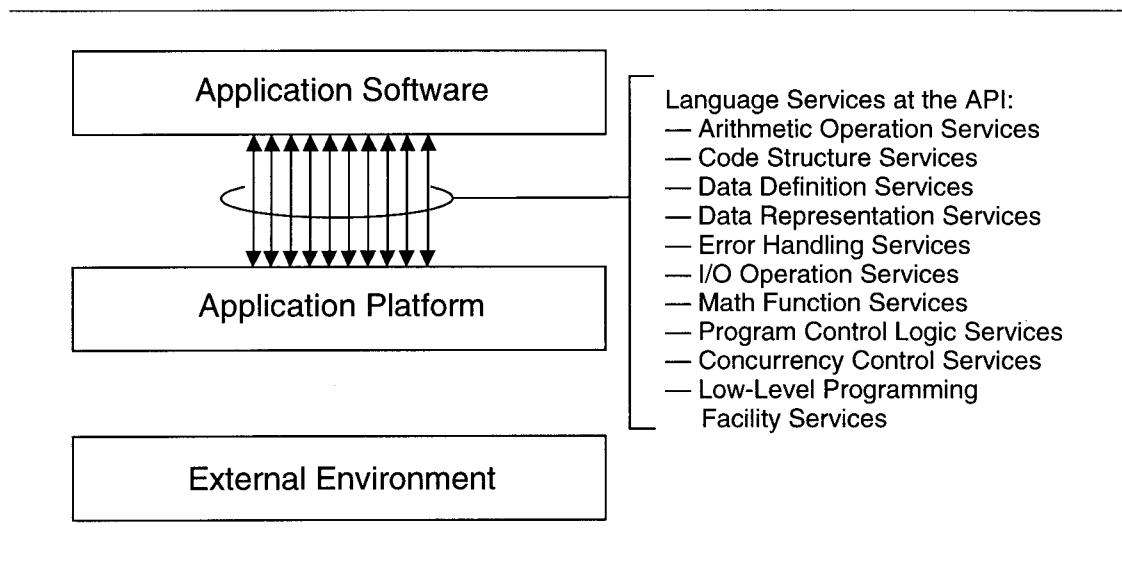


Figure 4.1—POSIX OSE Language Services Reference Model

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/IEC 9945-1 :1990 {68}.

4.1.5.1.8 Pascal

ISO/IEC 7185 :1990 {16} is the current version of the international standard for Pascal. The international standard for Extended Pascal is ISO/IEC 10206 :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.

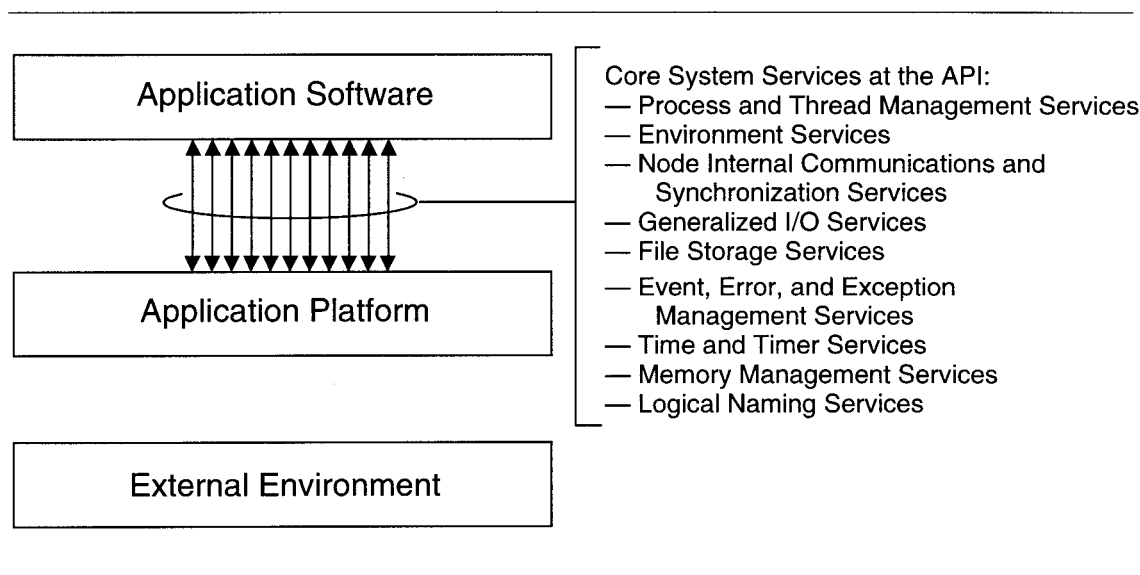


Figure 4.2—POSIX OSE Core System Services Reference Model

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.