--

--

IEEE P1003.0 Draft 14 – November 1991

STANDARDS PROJECT

Draft Guide to the POSIX Open Systems Environment

Sponsor

Technical Committee on Operating Systems and Application Environments

of the

IEEE Computer Society

Abstract: IEEE Std 1003.0-199x presents an overview of open system concepts and their application. Information is provided to persons evaluating systems 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 (OSE) requirements are not being addressed currently. Finally, the OSE profile concept is discussed with examples from several application domains.

Keywords: application portability, open system environments, profiles, POSIX

P1003.0 / D14
November 1991

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

November 1991 *SH XXXXX*

Editor's Notes

This section will not appear in the final document. It is used for editorial comments concerning this draft.

Comments in italics are not intended to form part of the final guide; they are editor's or coordinator comments for the benefit of reviewers.

This draft uses small numbers in the right margin in lieu of change bars. ‘‘E’’ denotes changes from Draft 13 to Draft 14. I have removed all old diff-marks from Drafts 3 through 13 to facilitate mock ballot review. Purely editorial changes such as grammar, spelling, cross references, or removals of editorial notes are not diff-marked. Unfortunately, it is not possible to accurately diff-mark the figures. Note that an empty line with a diff mark denotes deleted text. There are a large number of these in Draft 14. My convention is that I remove these empty lines in the next draft.

The references to standards and other documents are still awaiting a reasonably stable draft for a massive global update. I expect this may occur after the first round of official IEEE balloting. The ISO and IEEE style is to fully identify such documents in either the Normative Reference clause or the Bibliography; each entry contains the full title, the year of approval, and the current status (draft, CD, DIS, etc.). Elsewhere in the guide, a terse reference to the standard number is followed by the item number in the reference list, such as:

POSIX.1 {2}
ISO/IEC 10646 {B33}

A few titles have been modified in Section 4 to adhere to the template for the services clauses. These mostly affect the Standards, Specifications, and Gaps subclause and they are not diff-marked unless some significant text change accompanies them.

To make draft handling in the meetings easier, each significant clause is set up to print starting on a recto page. This means that there is a larger number of blank pages than in previous drafts (assuming that the copy room handled the print master correctly). Just doing our bit for deforestation . . .

Hal Jespersen

Editorial Contacts

Please send comments regarding the content and approach of this document to:

Fritz Schulz
Open Software Foundation
620 Herndon Parkway - Suite 200
Herndon, VA 22070
+1 (703) 481-9851
FAX: +1 (703) 437-0680
E-Mail: `fschulz@osf.ORG`

Please report typographical errors (and index suggestions) to:

Hal Jespersen
POSIX Software Group
447 Lakeview Way
Redwood City, CA 94062
+1 (415) 364-3410
FAX: +1 (415) 364-4498
E-Mail: `hlj@Posix.COM`

(Electronic mail is preferred.)

Online Access

This draft is available in various electronic forms to assist the review process. Our thanks to Andrew Hume of AT&T Bell Laboratories for providing online access facilities. Note that this is a limited experiment in providing online access; future drafts may be provided in other forms, such as diskettes or a bulletin board arrangement, but the instructions shown here are the only methods currently available. Please also observe the additional copyright restrictions that are described in the online files.

Assuming you have access to the Internet, the scenario is approximately

```
ftp research.att.com # research's IP address is 192.20.225.2
<login as netlib; password is your email address>
cd posix/p1003.0/d14
get toc index
binary
get p11-20.Z
```

The draft is available in several forms. The table of contents can be found in `toc`, pages containing a particular section are stored under the section number, sets of pages are stored in files with names of the form `p`*n*–*m*, and the entire draft is stored in `all`. By default, files are ASCII. A `.ps` suffix indicates PostScript. A `.Z` suffix indicates a `compress`'ed file. The file `index` contains a general description of the files available.

These files are also available via electronic mail by sending a message like

```
echo send 3.4 4.6 6.2 from posix/p1003.0/d14 |
        mail netlib@research.att.com
```

If you use email, you should *not* ask for the compressed version. For a more complete introduction to this form of *netlib*, send the message

```
send help
```

POSIX.0 Change History

This section is provided to track major changes between drafts. Since it was first added in Draft 10, earlier entries have been omitted.

Draft 14 [November 1991] First mock ballot. E

- Software Development clause 4.11 moved to Annex E. E
- *Other Details of Changes to be Provided* E

Draft 13 [September 1991]

- *To Be Provided*

Draft 12 [June 1991]

- Clause 4.9: Separated OLTP model discussion into two parts: the part consistent with the POSIX OSE Model; and the ''real world'' part dealing with System Integration Interfaces.
- Section 6: Further clarified ''base standard'' and ''profile'' definitions. Renamed profile ''types''.
- *Additional To Be Provided*

Draft 11 [March 1991]

- *To Be Provided*

  HLJ : I don't do this automatically because I don't know what issues you consider important. This [very brief] text should be provided by each Section Leader along with the regular submissions. It is meant to provide casual readers of the guide (such as in WG15, where they don't get every draft) with a broad overview of the big changes.

Draft 10 [December 1990]

- *To Be Provided*

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 the 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, the 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

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contents

PAGE

FIGURES

TABLES

Introduction

(This Introduction is not a normative part of P1003.0 Guide to the POSIX Open Systems Environment, but is included for information only.)

Purpose

There are many standards efforts going on throughout the world today. Standards are being developed in many areas of computing technology such as:

— Electrical Connectors

— Disk Interfaces

— Network Interfaces

— Application Program Interfaces

Each standards effort typically addresses a very small portion of the overall needs of an information processing system.

This guide brings together many different standards sufficient to address the scope of an entire information processing system. This combination of standards and specifications that are sufficient to address all of the user requirements of an information processing system is called an Open System Environment. This guide is not a base standard itself; it merely identifies standards that can be used when constructing a complete information processing system. Although this guide is a product of the IEEE POSIX standardization effort, its scope is much broader than the IEEE POSIX standardization 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 in the POSIX Open System Environment.

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

It may never be necessary to implement an information processing system that provides every standard in the POSIX Open System Environment. Typically, a subset of the POSIX Open System Environment is sufficient to satisfy the particular user requirements in each situation.

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

This guide is intended to be used by anyone interested in using standards in an information processing system, including: consumers, systems integrators, application developers, systems providers, and procurement agencies.

Taken as a whole, the guide maps existing and emerging standards onto the general requirements of a complete information processing system. In addition to listing and categorizing existing standards efforts, the guide identifies important requirements that standards efforts have not yet addressed.

The POSIX Open System Environment Reference Model

To describe the POSIX Open System Environment, the guide develops a reference model used to classify information processing standards. The reference model divides standards into two general categories:

Application Program Interface Standards

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

Platform External Interface Standards

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

These standards are very important because they allow a user to independently procure portions of their information processing systems from multiple vendors according to each user's needs.

In addition to these two interfaces identified in the model, there are other important interface between different computer system components: System Internal Interfaces. These interfaces have no direct impact on the external interface of a system or the application program interface to the system. System Internal Interfaces are beyond the direct scope of this guide because they do not directly impact application portability or system interoperability.

The services provided by the application platform are classified into four major categories:

— System services

— Communications services

— Information services

— Human-computer interaction services

Within these categories, services component areas are identified.

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

Goals

There are three goals of the POSIX OSE: portability, interoperability, and user portability. (While these terms are formally defined later in this guide and within various referenced standards, the following descriptions provide an overview of their meaning.)

Portability

Source Code Portability is accomplished through the use of the respective system/application interface standards and their extensions, thus allowing a user's application to operate on a wide range of systems. It is important to note that the aforementioned phrase ''wide range of systems'' connotes diverse hardware as well as software platforms.

Interoperability

Interoperability is characterized by the cooperative operation of applications resident on dissimilar computer systems. This cooperative operation is illustrated by data and functionality exchange.

User Portability

A consistent user interface allows users to move from system to system and between different applications on the same system with a minimum of retraining.

Benefits

The benefits derived in the use of the POSIX Open System Environment are real and quantifiable.

Simplified Vendor Mixing System Integration

As the standards for system integration and system interoperability are produced and implemented, the 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 their hardware based on the application needs rather than its ability to interoperate with their 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. When these standards are widely used there will be large pool of personnel who are familiar with working with the 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.

As the basic interfaces are developed and well defined, higher level standardized interfaces can be developed that add value to the basic interfaces. Using the higher level interfaces may speed development efforts.

Efficient Porting of Applications

The difficulty of moving an application from one hardware/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.

It is important to note that while standards-based systems enable applications to be ported between different systems, the standards do not guarantee that an application will be portable. Applications still must be properly engineered to ensure application portability.

Broadened Basis for Computer System Procurement Decisions

Computer users can now select and match hardware and software components from potentially different suppliers to fulfill an application requirement. This in turn allows decisions regarding computer systems procurements to be based less upon constraints imposed by incumbent vendors' products. The basis for competition will refocus on such factors as price, quality, value-added features, performance, and support. The stimulation of competition will benefit providers and users.

Related Standards Activities

The Standards Subcommittee of the IEEE Technical Committee on Operating Systems and Application Environments has authorized other standards activities that are related to the content of this guide.

The following table summarizes the current POSIX standardization efforts[1] and how they fit into this guide:

Project	Standard/Profile	Clause
P1003.1	System Interfaces	4.2
P1003.2	Shell and Utilities	4.9
P1003.3	Test Methods	
P1003.4	Realtime	4.2
P1003.5	Ada Bindings	4.2
P1003.6	Security	5.2
P1003.7	System Administration	5.3
P1003.8	Transparent File Access	4.3
P1003.9	Fortran Bindings	4.2
P1003.10	Supercomputing Profile	7.2
P1003.11	Transaction Processing Profile	7.2
P1003.12	Protocol-Independent Network Specification	4.3
P1003.13	Realtime Profile	7.2
P1003.14	Multiprocessing Profile	7.2
P1003.15	Batch System	4.9
P1003.16	C-Language Bindings	4.2
P1003.17	Directory/Name Services	4.3
P1003.18	POSIX Platform Profile	7.2
P1201.1	Human-Computer Interfaces	4.6
P1201.2	User Interface Drivability	4.6
P1224	X.400 API	4.3
P1237	RPC	4.3
P1238.0	FTAM API	4.3
P1238.1	OSI Networking API	4.3

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

Extensions are approved as ‘‘amendments’’ or ‘‘revisions’’ to this document, following the 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.

---

1) A *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; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are also available from this office.

If you have interest in participating in the TCOS 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, and ask to have this forwarded to the chairperson of the appropriate TCOS working group. If you have interest in participating in this work at the international level, contact your ISO/IEC national body.

P1003.0 was prepared by the 1003.0 working group, sponsored by the Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society. At the time this standard was approved, the membership of the 1003.0 working group was as follows:

Technical Committee on Operating Systems
and Application Environments (TCOS)

Chair: Jehan-François Pâris

TCOS Standards Subcommittee

Chair:	Jim Isaak
Vice Chairs:	Ralph Barker
	Robert Bismuth
	Hal Jespersen
	Lorraine Kevra
	Pete Meier
Treasurer:	Quin Hahn
Secretary:	Shane McCarron

1003.0 Working Group Officials

Chair:	Allen Hankinson
Vice Chair:	Kevin Lewis
Document Editor:	Hal Jespersen (sponsored by Mike Lambert)
Technical Editor:	Fritz Schulz
Secretary:	Charles Severance

Working Group

<Name to be provided> *<Name to be provided>* *<Name to be provided>*

The following persons were members of the 1003.0 Balloting Group that approved the standard for submission to the IEEE Standards Board:

<Name > <Institution> Institutional Representative

<Name to be provided>	*<Name to be provided>*	*<Name to be provided>*

When the IEEE Standards Board approved this standard on *<date to be provided>*, it had the following membership:

(to be pasted in by IEEE)

Guide to the POSIX Open Systems Environment

Section 1: General

Responsibility: Kevin Lewis

1.1 Scope

This guide identifies parameters for an open system environment using the POSIX operating system/application interface as the platform. These parameters are determined in three basic ways:

(1) By specifying building blocks identified as components

Currently these components are: system services, networking, human/computer interaction (HCI), graphics, system security and privacy, database, data interchange, and language requirements. This guide identifies the standards required within each component to achieve the goals of a POSIX open system.

(2) By identifying intra- and intercomponent issues

These issues involve the relationships that should exist between and among the different components. It is in the attempt to lay out and address these relationships that the concept of profiles (see 2.2.2 and Section 6) arises.

(3) By identifying voids

A void is determined by the absence, or lack of maturity, of formal standards development efforts. Voids may exist within available standards or may be an entire component. This guide provides assistance to those users who have already constructed, or plan to construct, profiles and to those users who currently use, or plan to use, profiles. The profile concept allows users to identify those standards that address their specific needs. The profile also serves to identify the need for future standards development in a specific area. This guide explains the manner in which these standards relate to each other.

1.2 Normative References

Note to reviewers: This clause is not complete. A list of referenced standards and other publications needs to be provided, contrasted against the list of interesting background documents that should go into the bibliography, included as Annex B. It currently consists only of sample entries. It will be replaced in a later draft.

The following standards contain provisions which, 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 this part of this International Standard 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.

{1} ISO 8859-1: 1987, *Information processing—8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1.*[1)]

{2} ISO/IEC 9945-1: 1990, *Information technology—Portable operating system interface (POSIX)—Part 1: System application programming interface (API) [C Language]*

1.3 Conformance

Not applicable.

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

1.4 Test Methods E

Not applicable. E

Section 2: Terminology and General Requirements

Responsibility: John Williams

2.1 Conventions

This guide uses the following typographic conventions:

— The *italic* font is used for cross references to defined terms within 1.3, 2.2.1, and 2.2.2.

In some cases tabular information is presented ''inline''; in others it is presented in a separately labeled Table. This arrangement was employed purely for ease of typesetting and there is no normative difference between these two cases.

The typographic conventions listed above are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this guide.

NOTEs provided as parts of labeled Tables and Figures are integral parts of this guide (normative). Footnotes and NOTEs within the body of the text are for information only (nonnormative).

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 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.

2.2.1.6 POSIX : The term ''POSIX'' has been evolving recently into a generally positive term with, unfortunately, a number of different meanings. This subclause attempts to define the word and some related terms. The intent is to insure that the term POSIX is used in a useful and predictable manner in this document.

As background, note that POSIX is sometimes used to denote the formal standard IEEE Std 1003.1-1990, sometimes to denote that standard plus related standards and drafts emerging from IEEE P1003.x working groups, and sometimes to denote the groups themselves. In all those cases, it should be noted, POSIX is used as a noun.

This document will use the term ''POSIX'' only as an adjective, and will use it only in well defined ways. This subclause serves as a preview of the usages in this book of POSIX terms. (These terms are defined, formally, or informally in subsequent clauses, and you will be referred to those clauses as appropriate.)

The original POSIX standard will be referred to by its name, ISO 9945, and not by the term POSIX.

The IEEE groups developing standards related to ISO 9945 are called, in this document, *POSIX working groups* . Examples are the IEEE working groups P1003.2, P1003.3, etc. The groups' names will be abbreviated POSIX. 2, POSIX. 3, etc.

The standards emerging out of the POSIX working groups will be referred to by their formal names (e.g., IEEE P1003.2 Draft 9) and are called either *POSIX Base Standards* or *POSIX Standardized Profiles* (POSIX SP s).

2.2.2 General Terms

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

2.2.2.1 application: The use of capabilities (services/facilities) provided by an information system specific to the satisfaction of a set of user requirements.

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

2.2.2.2 application platform: A set of resources that support the services on which an application or 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.

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

The application program interface is primarily in support of application portability, but system and application interoperability are also supported via the communications API.

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

2.2.2.5 Application Environment Profile (AEP): A profile, specifying a complete and coherent subset of the OSE, in which the standards, options, and parameters chosen are necessary to support a class of applications.

2.2.2.6 base standard: A standard or specification that is recognized as appropriate for normative reference in a profile by the body adopting that profile.

2.2.2.7 Communications Interface: The boundary between application software and the external environment, such as other application software, external data transport facilities, and devices.

The services provided are those whose protocol state, syntax, and format all must be standardized for interoperability.

2.2.2.8 External Environment Interface (EEI): The interface between the application platform and the external environment across which information is exchanged.

The External Environment Interface is defined primarily in support of system and application interoperability.

The primary services present at the External Environment Interface comprise:

- — Human/Computer Interaction Services
- — Information Services
- — Communications Services

2.2.2.9 external environment: A set of external entities to the application platform in which information is exchanged.

These devices include displays, disk drives, sensors, and effectors directly accessible within the system.

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

2.2.2.11 Human/Computer Interface: The boundary across which physical interaction between a human being and the application platform takes place.

2.2.2.12 Information Interchange Interface: The boundary across which external, persistent storage service is provided.

Only the format is required to be specified for data portability and interoperability.

2.2.2.13 interface: The shared boundary between two functional units, defined by functional characteristics and other characteristics, as appropriate.

2.2.2.14 internationalization: The process of designing and developing a product with a set of features, functions, and options intended to facilitate the adaptation of the product to satisfy a variety of cultural environments.

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

2.2.2.16 language-binding API: The interface between applications and application platforms based on language-independent binding APIs and consistent with the paradigms used for a specific programming language.

2.2.2.17 language-independent service specification: A specification that facilitates the management and development of consistent language-binding standards.

2.2.2.18 locale: A description of a cultural environment.

2.2.2.19 localization: The process of utilizing the internationalization features to create a version of the product for a specific culture.

2.2.2.20 local adaptation: The process of modifying a product that has hard-coded biases of one culture to the hard-coded biases of another culture.

2.2.2.21 open specifications: Public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards.

2.2.2.22 Open System Application Program Interface: A combination of standards-based interfaces specifying a complete interface between an application program and the underlying application platform.

This is divided into the following parts:

— Human/Computer Interaction Services API

— Information Services API

— Communications Services API

— System Services API

2.2.2.23 open system: A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software:

— to be ported with minimal changes across a wide range of systems

— to interoperate with other applications on local and remote systems

— to interact with users in a style that facilitates user portability.

2.2.2.24 Open System Environment (OSE): The 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.

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

2.2.2.26 performance evaluation: The technical assessment of a system or system component to determine how effectively operating objectives have been achieved.

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

2.2.2.28 portability: The ease with which software can be transferred from one information system to another.

2.2.2.29 POSIX Open System Environment (POSIX OSE): The Open System Environment in which the standards included are International, Regional, and National Information Technology Standards and profiles that are in accord with ISO/IEC 9945 (POSIX).

This guide represents the POSIX OSE as it existed when the guide was approved.

2.2.2.30 POSIX OSE Cross-Category Services: A set of tools and/or features that has a direct effect on the operation of one or more component of the POSIX Open System Environment, but is not in and of itself a stand-alone component.

2.2.2.31 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 POSIX.0 Reference Model for POSIX systems.

NOTE: Which POSIX base standards form the basis of the POSIX SPs is still open. Annex A discusses some of the issues involved.

2.2.2.32 process: An address space and single thread of control that executes within that address space, and its required system resources.

A process is created by another process issuing the *fork*() function. The process that issues *fork*() is known as the parent process, and the new process created by the *fork*() as the child process.

2.2.2.33 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, necessary for accomplishing a particular function.

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

2.2.2.35 protocol (OSI): A set of semantic and syntactic rules that determine the behavior of [OSI-] entities in the same layer in performing communication functions.

2.2.2.36 redirection: A system profile construction method of starting at a base platform and adding new services by allowing a service component to ask the base platform to redirect all requests for that type of service to the service component.

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

2.2.2.38 reference model: A simplified description or representation of something.

2.2.2.39 scaleability: The ease with which software can be transferred from one graduated series of application platforms to another.

E

2.2.2.40 security: The protection of computer hardware and software from accidental or malicious access, use, modification, destruction, or disclosure.

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

2.2.2.41 service delivery latency: The interval between (a) context switch from an application context to the operating system context, and (b) satisfaction of the service request.

2.2.2.42 service request latency: The interval between (a) context switch from an application context to the operating system context, and (b) the reverse context switch from the operating system context to the application context for a given service request.

2.2.2.43 software: The programs, procedures, rules, and any associated documentation pertaining to the operation of a data processing system.

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

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

2.2.2.46 standards: Documents, established by consensus and approved by a recognized body, that provide, for common and repeated use, rules, guidelines, or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context.

E

2.2.2.47 System Internal Interface (SII): The interface between application platform service components within that platform; it may be standardized or nonstandard.

2.2.2.48 system services: Firmware and software that provide an aggregation of network element functions into a higher level function; provide an interface to the data contained in the system.

2.2.2.49 System Services API: An interface providing access to services associated with the application's internal resources.

The System Services API has two parts: Language Specifications and Processing Services API.

2.2.2.50 system software: Application-independent software that supports the running of application software.

2.2.2.51 transaction: A unit of work consisting of an arbitrary number of individual operations all of which will complete successfully (or be of 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 a transaction completion.

2.2.2.52 transaction application program: A program written to meet the requirements of a chosen Transaction Processing (TP) application.

Such programs allow a sequence of operations that involve resources such as terminals and databases. The transaction AP specifies transaction boundaries. The transaction AP as defined here is a logical entity and may involve an arbitrary number of processes.

2.2.2.53 validation: The process of evaluating a ported application, software, or system to ensure compliance with requirements.

2.2.3 Abbreviations

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

2.2.3.1 API: Application Program Interface

2.2.3.2 EEI: External Environment Interface

2.2.3.3 POSIX. 0: This guide.

2.2.3.4 POSIX. *n*: An IEEE POSIX working group, where the number *n* represents the decimal notation in the IEEE P1003 series. Alternatively, when apparent from context, the latest standard issued, or under development, by that working group.

2.2.3.5 SII: System Internal Interface.

Section 3: POSIX Open System Environment

Responsibility: Fritz Schulz

The POSIX Open System Environment (OSE) is a collection of concepts that provide a context for user requirements 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 OSE concepts are also intended to be conventional within computer science. The intention is not to break new ground, but to establish a minimum and unambiguous terminology and set of concepts for identification and resolution of portability and interoperability issues.

The POSIX Open System Environment is defined in five parts:

(1) General requirements are identified that apply to the POSIX OSE as a whole 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 to identify interfaces across which service requirements should be satisfied. The elements are chosen to expose those interfaces that are significant to the profile writer or user.

(3) Using the interfaces identified in the reference models, each subclause 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.

Definition of the service requirements is not constrained by the availability of standards. Service requirements that are not currently satisfied via standards are discussed in either the Emerging Standards subclause, or under Gaps.

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 of the Section 4 clauses concludes with a discussion of standards that are related to the services.

(4) Section 5 discusses issues and requirements 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.

E

Definition of the service requirements is not constrained by the availability of standards. Services requirements that are not currently satisfied via standards are discussed in either the Emerging Standards subclause, or under Gaps.

3.1 POSIX Open System Environment — General Requirements

The POSIX Open System Environment should satisfy the requirements in the following list:

(1) Application Portability at the Source Code Level

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

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

(2) System Interoperability

The POSIX OSE shall enable application software and system service interoperability.

Rationale: Communications 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-OSE-based systems

(3) User Portability

The POSIX OSE shall enable human users to operate on a wide range of systems without retraining.

Rationale: Standard methods and services for supporting human/computer interaction are a key aspect of the definition of an open system (see Section 2). Elimination of gratuitous differences in the interface that the application platform presents to the user via standards is a significant aspect of this task.

(4) Accommodation of Standards

The POSIX OSE shall accommodate existing, imminent, and new information technology standards.

Rationale: If the POSIX OSE were constrained to current 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 must evolve as standards emerge and technology changes.

An inevitable tension exists between establishing fixed standards and providing for technology enhancement. Therefore, the POSIX OSE must be sufficiently general to allow for technology growth and yet specific enough to act as a guide for standards development.

(5) Accommodation of New Information System Technology

The POSIX OSE shall accommodate new Information System Technology.

Rationale: The POSIX OSE must strive to satisfy the full range of the users' functional requirements. This is undoubtedly a requirement that will only be fully realized over time, but it reflects the goal of the POSIX OSE.

(6) Application Platform Scalability

The POSIX OSE shall be scalable to platforms of varying power and implementation complexity.

Rationale: This reflects the realities of the potential users of the POSIX OSE. This requirement affects individual standards as well as the conditions under which various of the standards can or should be combined into profiles.

For example, where similar services are provided by both workstation type application platforms and supercomputers, 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.

(7) Distributed System Scalability

The POSIX OSE shall 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.

(8) Implementation Transparency

The POSIX OSE shall provide implementation technology transparency.

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.

(9) User's Functional Requirements

The POSIX OSE shall reflect the full scope of the user's functional requirements, within the context of the other requirements above.

Rationale: The POSIX OSE will provide the context within which application software portability can be addressed and it is the set of user's functional requirements that defines the scope of transportable service needs.

3.2 POSIX Open System Environment Reference Model

The POSIX OSE is based on a reference model with the full information system as its scope. As such, it spans the gap between requirement specification and the design of any specific information system. 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, system interoperability, and may encourage software reuse. It will certainly allow for more compact and correct procurement specifications.

The definition of this reference model is an engineering and management task and not a scientific one. There are many possible models and, while it might be interesting to contemplate an optimal one, a reference model that satisfies the requirements is all that is necessary.

An information system reference model must satisfy conflicting requirements similar to those encountered in traditional architectural disciplines. The reference model must be structured enough to encourage the generation and use of standards and standard components. Yet it must also be flexible enough to accommodate tailored and special purpose components necessary to meet realworld needs.

The POSIX OSE Reference Model is a set of concepts, interfaces, entities, and diagrams that provides a basis for specification of standards. The POSIX OSE Reference Model will provide guidance and direction for future standardization and integration efforts. In order for the POSIX OSE to evolve and mature, it will be 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 application platform user's perception (mental model) of the overall large distributed system used to support the user enterprise. This point of view will assure that the:

— Information technology users will have the proper services to meet their requirements, and

— Information technology vendor implementations will not be constrained unnecessarily.

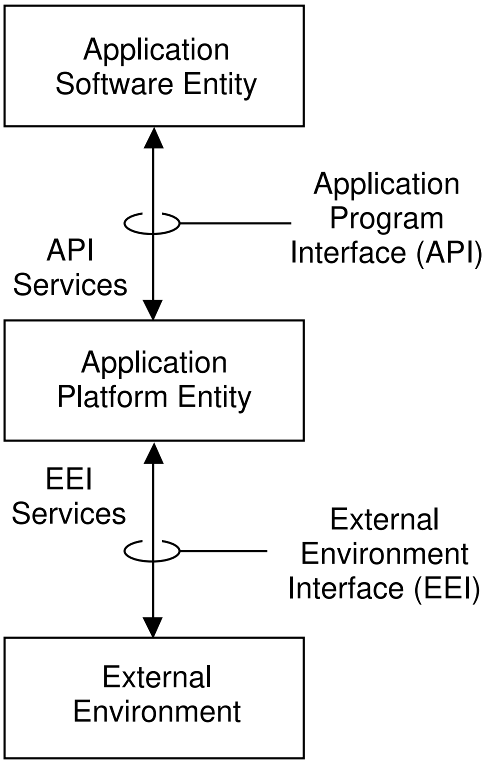

Figure 3-1 – POSIX OSE Reference Model

Figure 3-1 depicts the basic elements of the POSIX Open System Environment Reference Model. These include three entities (Application Software, Application Platform, and External Environment) and two interfaces between them, identified as the Application Program Interface (API) and the External Environment Interface (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. More detailed requirements exist for each service category described in Section 4. The service specification has been defined to be robust and 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.

Standards (in the sense of formally adopted consensus specifications) address only interfaces between entities, as well as 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 must be properly engineered with an explicit portability objective in order to achieve it.

The Reference Model is not a layered model. The application platform provides services to a variety of users across both platform interfaces. A human being invokes the platform services at the External Environment Interface. If an application developer is the application platform user, the services offered at the application program interface (API) are invoked at the source code level.

All of these features may be available locally or remotely if the system is connected to a larger distributed system. All other resources and objects 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 expands Figure 3-1 to identify elements of the Reference Model entities. For the purposes of this discussion, the term ''entities'' will be used when discussing the classification of items (i.e., ''things'') related to application portability. The term ''component'' will only be used when an entity is further decomposed into constituent parts. The application software entity is the only entity that is decomposed into components.

Application Software is defined (see 2.2.2.4) as software specific to an application. It is composed of:

— Programs (source code, command/script files, etc.)

— Data (user data, application parameters, screen definitions, etc.), and

— Documentation (online documentation only; hardcopy not included).

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

An application program may be divided into two parts:

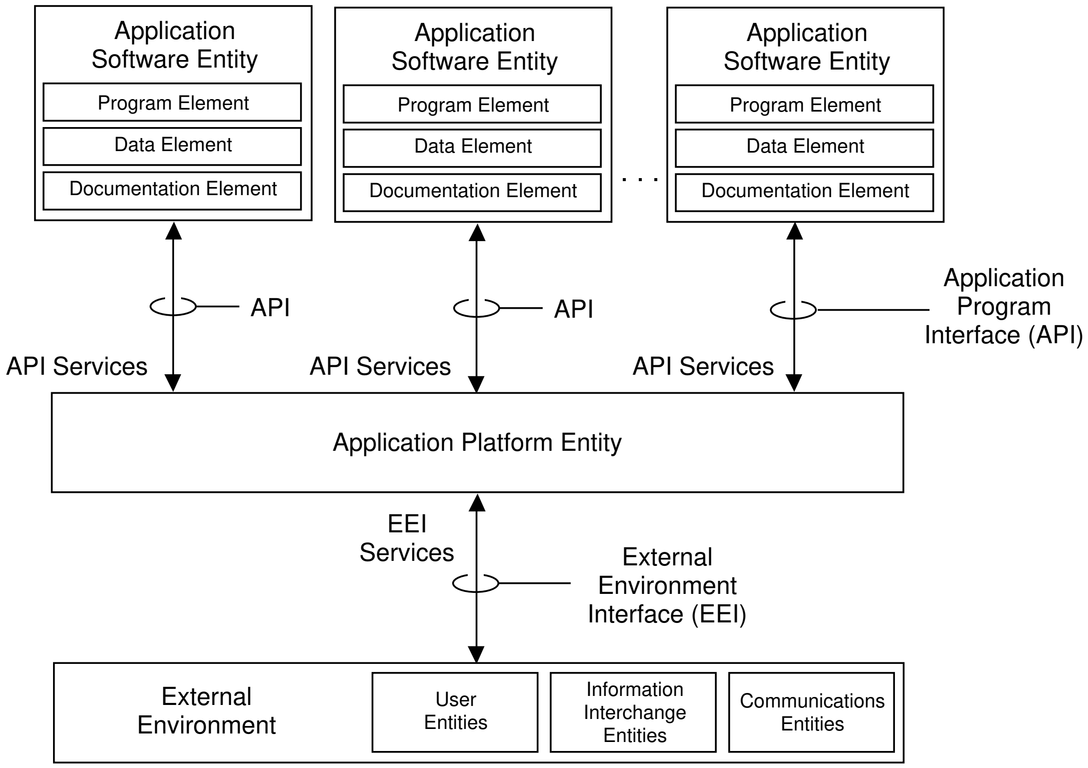

Figure 3-2 – POSIX OSE Reference Model — Entities

— An *invariant* portion of source code, requiring no change when ported, and

— A *variant* portion of source code, which requires changes when ported.

The objective of any effective application software portability method 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 portability-standard compliant) system and run without source code modification. However, since standards exist for which strictly conforming application software requires modification (e.g., memory requirements, processor-specific COBOL statements), this can only be approximated.

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 communications mechanisms.

The Application Platform is defined (see 2.2.2.2) 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 must access all resources via service requests across the API. Examples of application platform elements could include an operating system kernel, a realtime monitor program, and all hardware and peripheral drivers.

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 for systems built to the POSIX OSE will differ greatly depending upon the requirements of the 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 communications entities.

Human users are not further classified, but are treated as an abstract, or average, person. Information interchange entities include removable disk packs, floppy disks, and security badges. Communications entities include phone lines, local area networks, and packet switching equipment

3.2.2 Reference Model Interfaces

Figure 3-3 expands Figure 3-1 to identify the services available at the reference model interfaces.

Between these three classes of entities there are two types of interface where standards and other open system specifications are required to enable application software portability and interoperability. These two interface types are labeled as the Application Program Interface (API) and the External Environment Interface (EEI).

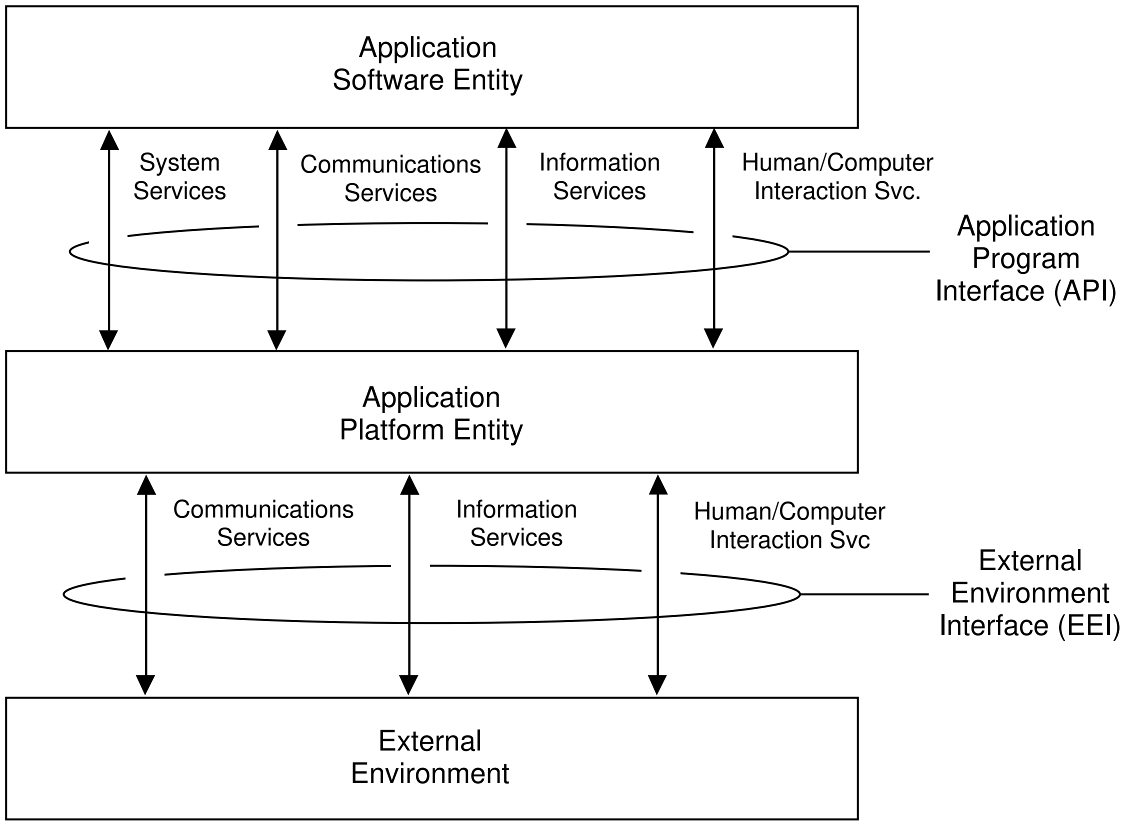

Figure 3-3 – POSIX OSE Reference Model — Interfaces

3.2.2.1 External Environment Interface (EEI)

The External Environment Interface is defined (see 2.2.2.8) 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 also is indirectly supported by reference to common concepts linking specifications at both interfaces. The services available at the EEI comprise:

— Human/Computer Interaction Services

— Information Services

— Communications Services

The Human/Computer Interaction EEI is the boundary across which physical interaction between the human being and the application platform takes place. Examples of this type of interface include CRT displays, keyboards, mice, and audio input/output devices. Standardization at this interface will allow users to access the services of compliant systems without costly retraining.

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

The Communications Services EEI 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 must be standardized for application interoperability.

3.2.2.2 Application Program Interface (API)

The Application Program Interface (API) is defined (see 2.2.2.3) as the interface between the application software and the application platform across which all services are provided. It is defined primarily in support of application portability, but system and application software interoperability also are supported via the communications services API.

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 specifies a complete interface between the application software and the underlying application platform, and may be divided into the following parts:

— System Services API

— Communications Services API

— Information Services API

— Human/Computer Interaction Services API

The last three APIs listed are required to provide the application software with access to services associated with each of the external environment entities.

The first API is required to provide access to services associated with the application platform internal resources, identified as the System Services API. This interface may be divided into two types of specifications; i.e., Language Service and System Services API specifications.

Definitions of services at the API 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:

(1) Those traditionally associated with the language specifications, such as program control (if . . . then . . . else), math functions, string manipulation, etc., defined as *the programming language API*, and

(2) Services provided by the underlying application platform defined independent of language, such as interprocess communications,

interobject messages, access to the user interface, and data storage. Specifications of for these services are defined independently of any programming language, and are identified as *language-independent service specifications* .

(3) The 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 APIs*.

Creation of a *language-independent service specification* facilitates 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., POSIX.1 {2}, MOSI, etc.) are being translated into ''language-independent'' specifications, with one or more bindings for specific languages.

3.2.3 EEI-API Service Relationships

The relationships between similarly named services provided at the API and 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, and can be thought of as having been ''translated'' into, communication services provided at the EEI.

Fortunately, it is not essential for the purpose of satisfying the requirements of the POSIX OSE to specify these relationships in detail. In fact, a detailed 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 network external to the platforms, but connected to them via the communications EEI, as in Figure 3-4. For an application software entity to gain access to the EEI services, communications services are requested at the API. The implementation of the application platform translates these API requests into appropriate action at the EEI.

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

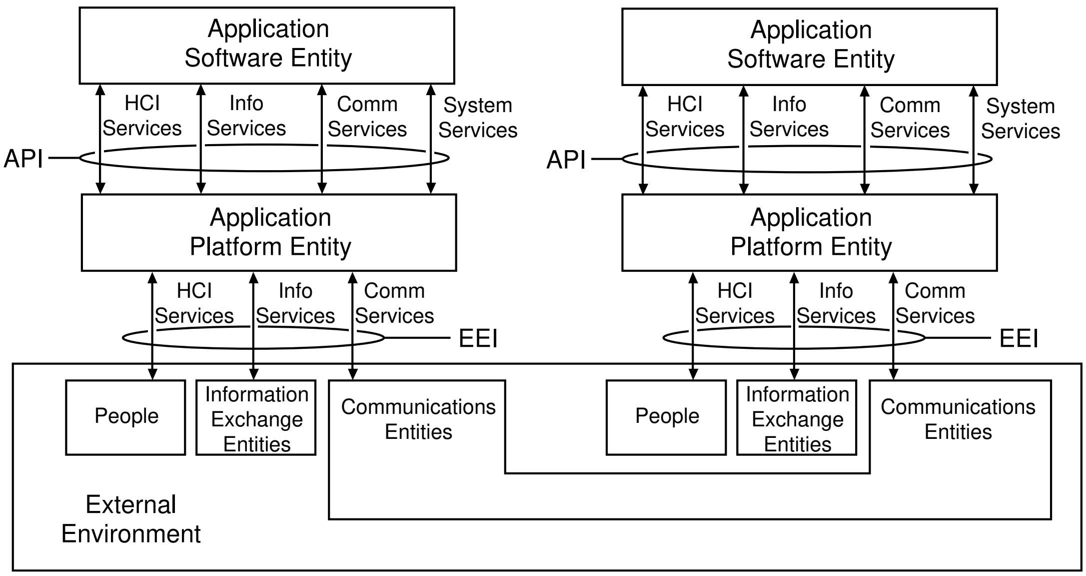

Figure 3-4 – POSIX OSE Reference Model — Distributed Systems

services via the network.

Distributed Systems are manifest in this model primarily through the use of the distributed system network services API. As can be seen in Figure 3-5, distributed systems are a refinement of the POSIX Network Environment Model shown in Figure 4-3. As such, a perceived Application Platform may in fact be comprised of several (or many) individual application platforms. However, in the distributed environment, they operate and are viewed as a single entity by the using applications. Within this extended application platform are the embedded network services necessary for the elements of a distributed environment to function.

Within the distributed environment, network access between the platforms that make up the ‘‘perceived’’ application platform are handled using the Distributed Systems Network Services APIs. Network services for access between ‘‘perceived’’ application platforms will use the Network Services EEI between the platforms.

3.3 POSIX Open System Environment Services

This guide defines a uniform set of standard 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 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 OSE reference model (see 3.2) to provide context for service

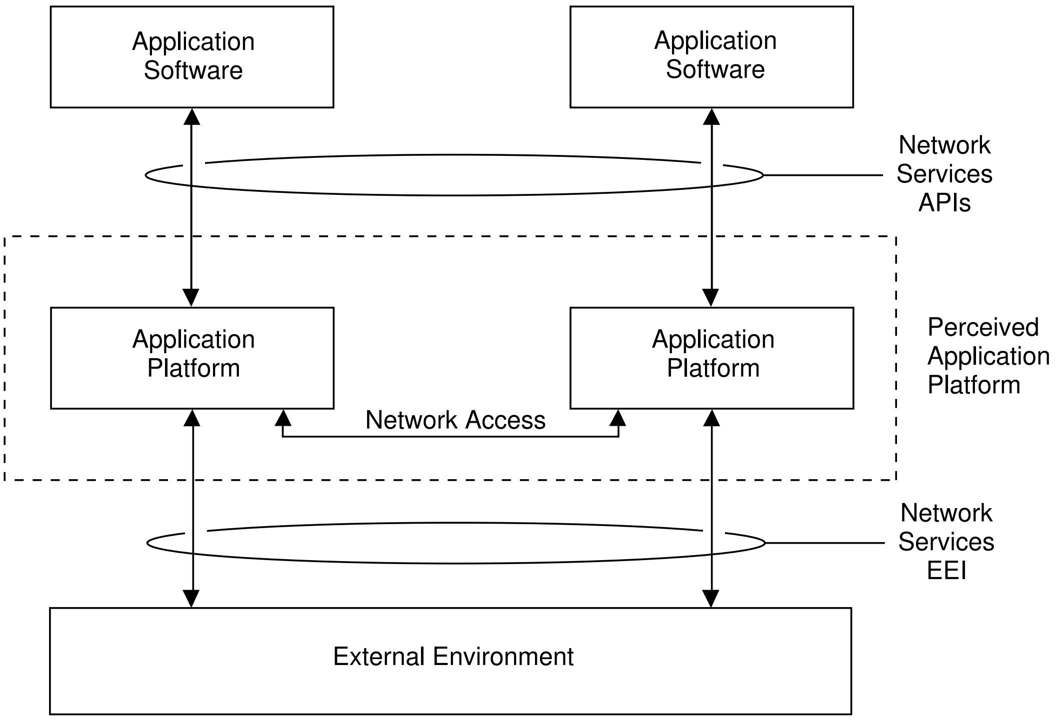

Figure 3-5 – Distributed System Environment Model

specification. Services and associated standards are then defined for each category. Finally, POSIX OSE Cross-Category Services affecting each category are discussed.

The service descriptions for each category are intended to be complete and not merely representative. Further refinement through successive releases of this document will lead to a complete specification.

3.4 POSIX Open System Environment 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 by the application platform user. The factors used to select standards will be described followed by the selection precedence.

Note that while the services are stated with a clear partitioning in mind, the standards reflect the current 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

The selection criteria for standards to be included in the POSIX OSE are based upon four concepts. Those concepts are Those concepts are openness, Stage of Completion, stability, Geographic Scope of Consensus, Functional Scope Addressed within this guide, Consistency with POSIX.1 {2}, and Availability for Unencumbered Implementation.

(1) Openness

Standards development organizations can differ from one another by virtue of their ‘‘openness.’’ That is, some standards development bodies utilize an open forum for the development of standards while other bodies use a closed 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 standards developed by bodies using a closed forum.

(2) Stage of Completion

Another factor involved in the selection of standards for inclusion in the POSIX OSE is ‘‘stage of completion.’’ That is, 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 were made among standards, the more complete standards were favored.

(3) Stability

A third factor in determining which standards are included in the POSIX OSE is stability. This factor 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.

(4) Geographic Scope of Consensus

There are differences among standards development bodies with respect to the scope of their geographic consensus. Some among those bodies are formal standards bodies (i.e., accredited as standards developers by a recognized body). It is typical for those bodies to be authorized to develop standards for a particular technical topic and have their standards applicable to some defined geographic area. Formal standards development bodies are typically empowered to develop standards for either international, regional or national standards coverage.

The general rule applied in the selection of standards for inclusion in the POSIX Open System Environment 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 developed standards.

(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 per set of services.

(6) Consistency with POSIX.1 {2}

Standards listed in this guide are suitable for inclusion in a profile with POSIX.1 {2}, and do not contradict that standard in any way.

(7) Availability for Unencumbered Implementation

A standard or specification is listed only if it is available for implementation to the specification and distribution of that implementation is unencumbered. The specification qualifies for inclusion in the guide even if the document itself is a salable item.

3.4.2 Selection Precedence

The list below shows the precedence of standards and specifications as used for inclusion in the POSIX OSE. The order from top to bottom is from most to least preferred.

(1) Approved standards developed by accredited international bodies

(2) Approved standards developed by accredited regional bodies

(3) Approved standards developed by accredited national bodies

(4) Draft standards developed by accredited international bodies

(5) Draft standards developed by accredited regional bodies

(6) Draft standards developed by accredited national bodies.

(7) Recognized de facto standards and specifications developed by nonaccredited bodies using an open forum

(8) Approved standards and specifications developed by nonaccredited international standards bodies using a closed forum

(9) Approved standards and specifications developed by nonaccredited national standards bodies using a closed forum.

Standards projects for which there is no draft or approved standard are never selected for inclusion in the POSIX OSE.

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

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

3.5 POSIX Open System Environment Profiles

The results of Open System specification projects are collected into 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 (or instance) of system or set of application software. Profiles satisfy the requirements of application ‘‘domains’’ such as office or industrial automation, transaction processing, or real-time control systems.

This framework provides a way to characterize the functionality of profile activities. The current OSI profiles tend to focus strictly on the communications EEI. Other profiles might focus on a single component or span multiple interface types.

3.6 Application Platform Implementation Considerations

Profile writers need to be aware that in an open system environment, the application platform can be decomposed into independently procurable components. While standards are interface specifications, and as such are independent of implementation, there are aspects of platform implementation or construction that may affect the specification of standards, and that profile writers may want to address.

For each case, the portion of the application platform that implements any particular independently procurable service is described as the service component. Figure 3-6 shows an application platform made up of several service components. If components interact, the specification of the interface between service components within the application platform may be standardized or nonstandard (including proprietary).

An intercomponent interface is labeled in Figure 3-6 as ‘‘System Internal Interface’’ because it may be used to assemble an application platform from multiple components. Figure 3-6 shows how a System Internal Interface is shown in the reference model.

A standards-based SII between the application platform service components addresses portability and interoperability of the application platform service components, not portability and interoperability of application software and systems.

Development of an SII would also require a consensus to emerge on the ‘‘best’’ design and implementation of system software/hardware. Very little consensus has developed on the partitioning of the platform into components and consequent allocation of function to each. In fact, this aspect of system design has been in a constant and accelerating state of innovation for decades. One of the major objectives of the API is to provide a more stable interface that decouples application software from the constantly changing platform. This enables the migration of application software to platforms based on constantly upgraded technology. (See 3.1 ‘‘Accommodation of New Information System Technology’’.)

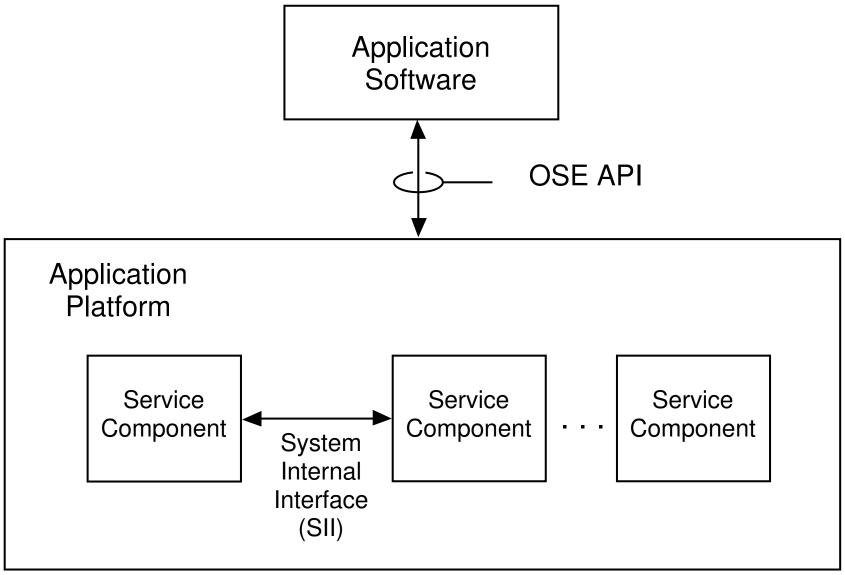

Figure 3-6 – Service Components and Interfaces

The relationship and services exchanged among the components may be quite complex and varied in different implementations. This complexity and variety would, of necessity, be reflected in an SII. It would not, however, be visible to the application software at the API, since one of the major objectives of the API is to hide this complexity. (See 3.1 ‘‘Implementation Transparency’’.)

Since SII specifications

— do not affect application portability and interoperability, and

— do not affect specification of the API and EEI, and

— are primarily driven by specific implementations of the application platform,

SII specification is beyond the scope of this guide.

Specification of SII 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 EEI.

There are a number of ways which the Application Platform can be divided into separate service components. The main decomposition methods are division, layering, and redirection. These methods are indistinguishable to the application software and external entities, in that they all interface to the application platform via the API and EEI, respectively. They assume a starting base application platform, which provides a subset of the required services.

3.6.1 Subdivision

In this commonly used method, the application platform is simply subdivided into a base and one or more service components. See Figure 3-7.

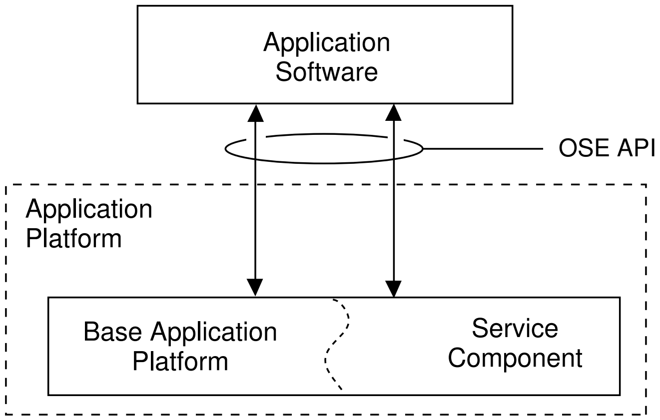

Figure 3-7 – Application Platform Implementation — Subdivision

One possible implementation of this is to link the appropriate service modules directly into the system kernel.

The internal interfaces used in this method are normally proprietary, and hence normally imply that both components will come from the same vendor.

In this case the Application Platform and the Application Platform Base are the same entity.

3.6.2 Layering

In layering, the service is interposed as a layer between the application software and the base application platform. See Figure 3-8.

This is the most common method of supplying a service component that is independent of the base. One possible implementation is to provide the service component as a set of library routines.

Whether the interface between the service layer and the base application platform conforms to any standards affects the portability of the service component. Note that specifying a standard API for this interface guarantees only that this component will be portable at the source level.

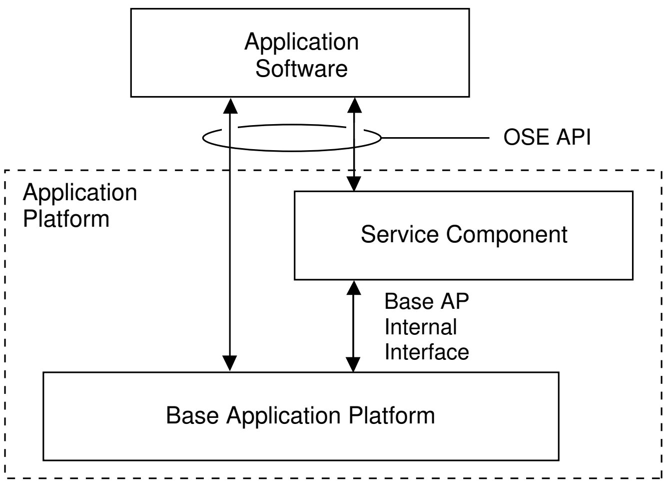

Figure 3-8 – Application Platform Decomposition II — Layering

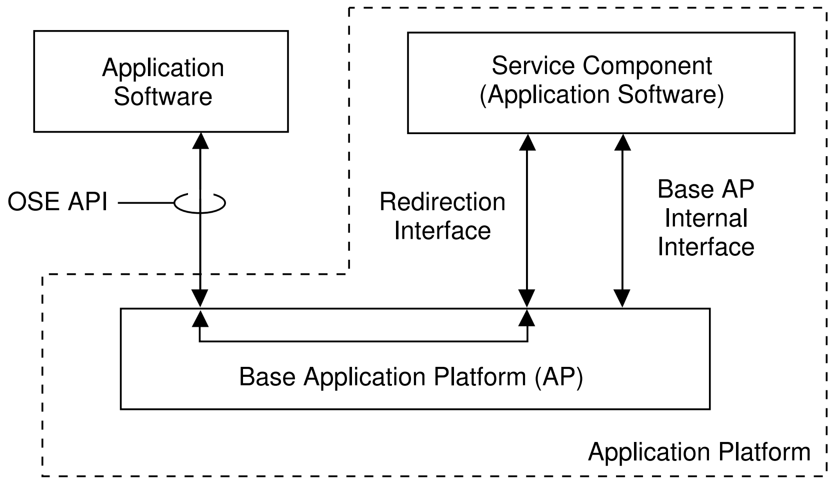

Figure 3-9 – Application Platform Decomposition III — Redirection

3.6.3 Redirection

Redirection allows a service component to ask the base application platform to redirect all requests for that type of service to the service component. See Figure 3-9. Possible examples of such services are device drivers, network protocol handlers, and database engines.

In actual implementation, the service component may or may not be a separate process. Possible implementations are: dynamically loadable kernel modules, library routines layered over IPC, and lightweight kernel processes.

Note that there are three interfaces. The application software normally sees a complete, standard API to the base. The service component has two interfaces—one to effect the redirection, and one to provide base services to the service application software entity. Considerations for portability discussed under Layering also apply here.

Note also that no POSIX standardization activity currently exists for the redirection interface.

Section 4: POSIX Open System Environment Services

Responsibility: *Fritz Schulz*

This section describes the services required in support of the objectives identified in this guide. The services are grouped in major categories defined in Section 3, with more detailed breakdowns within each category as appropriate. These categories are:

- System Services
  - 4.1 Language Services
  - 4.2 System Services
- Communications Services
  - 4.3 Network Services
- Information Services
  - 4.4 Database Services
  - 4.5 Data Interchange Services
  - 4.6 Transaction Processing Services
- Human-Computer Interaction Services
  - 4.7 Windowing System Services
  - 4.8 Graphic Services
  - 4.9 Character-Based User Interface Services
  - 4.10 User Command Interface 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 clause 3.2 and gives additional detail related to the interfaces and services discussed there. An optional subclause may discuss implementation considerations, similar to the discussion of 3.6.

4.*n*.4 Service Requirements

This text provides the definition of service requirements 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 service requirements 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 comprehensive for the area covered by the 4.*n*.4 requirements; there are no applicable standards or emerging standards excluded from the POSIX OSE. Within the table, the Type column refers to the status of the requirement:

S A current standard

E An emerging standard

G A requirement not satified by a formal standard (gap)

4.*n*.5.1 Current Standards

The following subclauses cite existing specifications that have been approved as standards by accredited standards bodies, in the order of precedence identified in 3.4.2. When service requirements are satisfied at a higher precedence level, specifications at a lower level are not listed.

4.*n*.5.2 Emerging Standards

The following subclauses provide 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 to be discussed in this subclause. It is expected that documents will migrate from 4.*n*.5.2 to 4.*n*.5.1 as they complete the consensus process.

4.*n*.5.3 Gaps in Available Standards

This subclause identifies those service requirements that have not been satisfied by existing or emerging standards. If all service requirements in this category have been met by existing or emerging standards, this subclause will be empty. Text in this subclause will be minimal.

4.*n*.5.3.1 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) without restriction, including all government and de facto standards.

4.*n*.5.3.2 Unsatisfied Service Requirements

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 any discussion of the Cross-Category Services in Section 5 that is specific to subclause 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.

Specification of performance metrics is not within the scope of this guide. E

4.1 Language Services

Responsibility: Don Folland

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.

Those responsible for system or software development will wish to write programs in code supported by an international standard and compile the code using a compiler that has a certificate of conformance issued by an accredited test center. Noncompliant extensions must be avoided if applications portability is to be maintained. Compilers should identify nonstandard-compliant code.

The languages that have been identified in this document are those seen to be in most popular use today for software development. The POSIX.2 shell command language is discussed in 4.10. 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 third-generation computer languages in use today for the development of applications; i.e., the languages used to write application programs. Fourth-generation languages are not currently addressed in this guide. 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 based on the reference model in Figure 3-1 is illustrated in Figure 4-1, but because the language services directly support the binding of the applications to the API, there is no EEI. However, the EEI is shown in Figure 4-1 for consistency.

At the simplistic level, the programmer developing an application that requires only basic operating system services will use a compiler that meets both the fundamental language standard (e.g., ISO 1989: 1985 for COBOL, ISO 1359: 1990 for Fortran) and the binding established for the relevant system calls in POSIX.1 {2}.

As identified in 4.6, an application program may also require database services that will be provided by the Database Manager API. The database vendor will

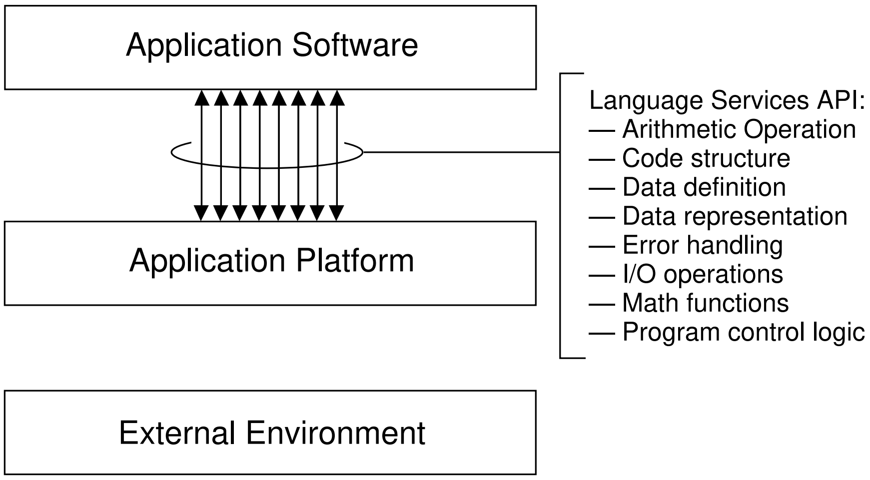

Figure 4-1 – Language Service Reference Model

offer an API to meet the requirements for the popular programming languages.

In a POSIX Open System Environment the intention is that support is provided for all languages identified in 4.1.4.

4.1.4 Service Requirements

Programming language services provide the basic syntax and semantic definition for use by a software developer to describe the desired application software function. While most clauses in this guide provide a comprehensive list of services, in the case of languages many services are a unique function of the language specification. Rather than extend the size of this guide, the detail is more appropriately found in the relevant language manuals and supporting standards.

4.1.4.1 Application Program Services

Programmers require the ability to write and execute a program in the language E
of their choice. The selection of a particular programming language for the development of an application may depend on a variety of factors, including the capability to provide some of the functions listed here:

— Arithmetic operation

— Code structure

— Data definition

— Data representation

— Error handling

— I/O operations

— Mathematical functions

— Program control logic

The programming languages identified in this clause are:

Ada
APL
BASIC
C
C++
COBOL
Common LISP
FORTRAN
Pascal
PL/1
Prolog

As well as making reference to the relevant language standard, where a programmer requires to call other services, e.g., seeks access to graphics kernel system, it will be necessary to refer to the relevant language binding to those services. Language bindings are identified in the Standards subclause, 4.*n*.10, of each service clause in Chapter 4.

4.1.4.1.1 Ada

Ada is a procedural language based on the Pascal programming language. It is capable of processing both numerical and textual data and has the key attributes of:

— Strong data typing

— Data abstraction

— Structured constructs

— Multitasking

— Concurrent processing

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

4.1.4.1.3 BASIC

BASIC is an interactive and procedural language with 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. BASIC offers:

— Conversational statements

— Free style input

— Segmentation of complex statements

— Six significant digits of accuracy

— 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 to manipulate the data, and so direct these objects to apply the necessary operations which 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 non-technical writers with a basic appreciation of information processing. The language offers file organization features, variable data length, input/output procedures, and report generation.

4.1.4.1.7 Common LISP

LISP is an interactive nonprocedural 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 variable length and dynamically adjusted; the items can be of different type.

4.1.4.1.8 FORTRAN

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

4.1.4.1.9 Pascal

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

4.1.4.1.10 PL/1

This 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.

4.1.4.1.11 Prolog

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

4.1.4.2 External Environment Interface Services

Not applicable.

4.1.4.3 Interapplication Software Entity Services

Not applicable.

4.1.4.4 Language Resource Management Services

Not applicable.

4.1.5 Standards, Specifications, and Gaps

4.1.5.1 Current Standards E

See Table 4-1. E

Table 4-1 – Language Standards

Service	Type	Specification	Subclause	E
Ada	S	ISO 8652	4.1.5.1	E
APL	S	ISO 8485	4.1.5.1	E
BASIC	S	ISO 6373	4.1.5.1	E
C	S	ISO/IEC 9899	4.1.5.1	E
C++	E	n/a	4.1.5.2	E
COBOL	S	ISO 1989	4.1.5.1	E
Common LISP	G	n/a	4.1.5.1	E
FORTRAN	S	ISO 1539	4.1.5.3	E
Pascal	S	ISO 7185	4.1.5.1	E
PL/1	S	ISO 6160	4.1.5.1	E
PL/1 (GP Subset)	S	ISO 6522	4.1.5.1	E
PROLOG	G	n/a	4.1.5.3	E

Ada

ISO 8652: 1987 is the current version of the international standard for Ada, which was an endorsement of the ANSI standard 1815A-1983.

APL

ISO 8485 is the current version of the international standard for APL.

BASIC

ISO 6373: 1984 is the current version of the international standard for minimal BASIC.

C

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

COBOL

ISO 1989: 1985 is the latest version of the international standard for COBOL, which was an endorsement of the ANSI standard X3.23-1985. An Addendum is in process at present entitled ‘‘Intrinsic function module.’’

Fortran

ISO 1539: 1990 is the latest revision of the international standard for Fortran.

Pascal

ISO 7185: 1983 is the current version of the international standard for Pascal, which was an endorsement of the British standard BS 6192-1982.

PL/1

ISO 6160: 1979 is the current version of the international standard for PL/1, which was an endorsement of the ANSI standard X3.53-1976. ISO 6522: 1985 is the current version of the international standard for a General Purpose subset of PL/1, which is an endorsement of ANSI standard X3.74-1981. A revision of this standard is at Draft IS stage.

4.1.5.2 Emerging Standards

BASIC

CD 10279 is a proposal for Full BASIC.

C++

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

Pascal

DIS 10206 is a draft international standard for extended Pascal.

4.1.5.3 Gaps in Available Standards

4.1.5.3.1 Standards and Specifications outside the POSIX OSE

None.

4.1.5.3.2 Unsatisfied Service Requirements

There is a requirement for standardization of the following languages:

- C++
- LISP
- Prolog

4.1.6 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, Database. Reference to the particular language binding standard is to be found in the relevant service clause.

4.1.8 Open Issues

While there are occasional calls for 4GL standards, there has been little effort applied so far.

4.2 System Services

Responsibility: Patricia Oberndorf

4.2.1 Overview and Rationale

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

System services are a key component of the application platform and represent the focus of the IEEE effort to produce POSIX base standards. A common set of system services provides support for the portability and the interoperability of application software. While other common services can aid application reuse, system services are those that are common to the largest number of applications.

4.2.2 Scope

System services cover those features that users have come to expect from operating systems or executives. They cover the areas of process management, file management, input/output, memory management, and print spoolers. Because there is 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, logical device drivers, and system initialization/reinitialization are included. Services related to distributed systems are also discussed, since application software sees these capabilities through the platform.

4.2.3 Reference Model

This subclause identifies the entities and interfaces specific to the system services of the POSIX OSE. The reference model presented here is consistent with and expands upon the reference model of Section 3. It provides the context for the discussion of System Services in this clause. The basis System Services model is shown in Figure 4-2.

This clause describes the system services portion of the application platform as viewed by a software developer (not necessarily the viewpoint of the end user). This view corresponds to the program design level of abstraction.

The system services API provides the interface between the application software and the system services from the source code point of view. The API defines the program designer's means of accessing the functions, objects, and services of the system.

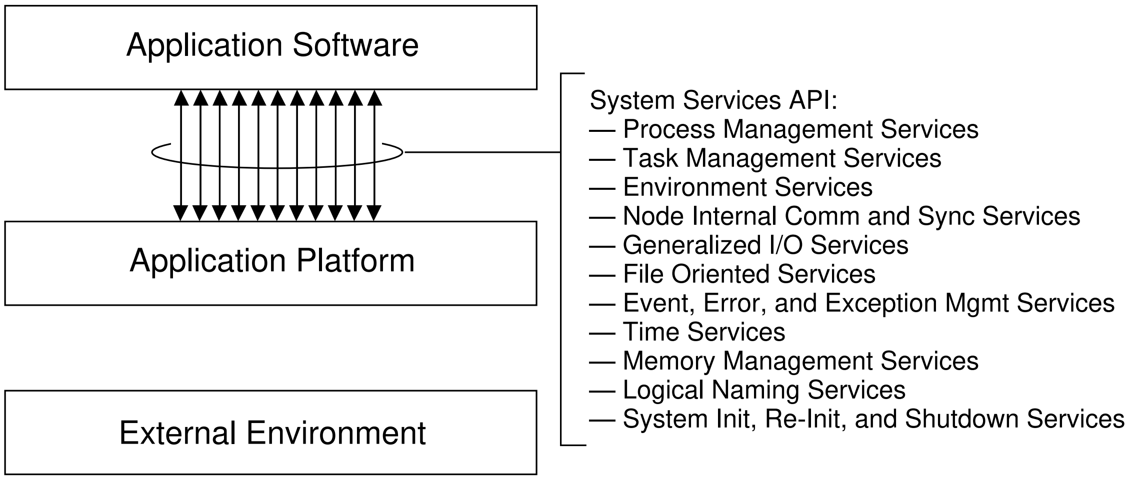

Figure 4-2 – System Services Reference Model

In order for the platform to protect system integrity and ensure system database consistency, application software competing for system resources must access all system resources via system service requests. The formal definition of these requests (or system calls) defines the system services portion of the API.

All of the system services may be available locally or remotely. Some of the system services may be performed remotely if the system is a distributed system with multiple processor nodes. Such distribution is not reflected in Figure 4-2 because it is transparent to users of the System Services.

The platform's device drivers and other software entities are seen as being available to an application program via invocation of the system services. Local devices include sensors, effectors, and connections to independent computing systems. The local devices themselves are a part of the external entities element of the system services reference model. The interfaces used by the application software are the logical device interfaces and are part of the system services. It should be noted that, even though the device drivers are represented within the system services portion of the application platform and the devices themselves are represented within the external entities, there is no unique system service interface illustrated at the EEI in Figure 3-3. This is not an oversight; such interfaces are not within the scope of this guide.

4.2.4 Service Requirements

This subclause identifies those processor-oriented system services required to support application portability and system interoperability. Subclause 4.2.4.1 describes those system services directly available to an application program via the System Services API. Other processor-oriented services are described in 4.2.4.4. Subclause 4.2.5 identifies the applicable standards.

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

4.2.4.1 Application Program Interface Services

This subclause describes the major categories of system services available at the System Services API. These services include:

— Process Management Services

— Task Management Services

— Environment Services

— Node Internal Communication and Synchronization Services

— Generalized Input/Output Services

— File Oriented Services

— Event, Error, and Exception Management Services

— Time Services

— Memory Management Services

— Logical Naming Services

— System Initialization, Reinitialization, and Shutdown Services

4.2.4.1.1 Process Management Services

These services relate to the creation, management, and deletion of processes executing within the scope of an operating system. These processes are distinguished from ‘‘tasks’’ via the following characteristics:

— They have a single thread of execution per address space.

— There is substantial overhead for context switches.

— Specific attributes are associated only with processes.

In this context, ‘‘management’’ consists of those services that affect the execution of a process:

— Stop and restart execution of a process (e.g., suspend, resume)

— Modify processor allocation to a process (e.g., priority, timeslice)

— Modify scheduling of the process based on timer (or other) events

— Protect the process from interruption during critical periods

— Create a process and make it ready for execution

— Destroy a process and recover its resources

— Evaluate a reference to a process

— Evaluate a connection to a process, where a connection is a logical communication path between any two processes

These services schedule or arbitrate the usage of various resources of the OS, particularly the central processing unit (CPU). The scheduling services must be able to queue up requests to use a particular resource. This situation is made more complicated by the common need to schedule processes to run cyclically at a fixed period. When a resource becomes idle, the scheduler must select one of the ‘‘requesters’’ of the resource to grant use of the resource. These services are listed separately rather than under the services that use scheduling to emphasize that there should be uniformity and consistency of scheduling across the range of resources.

Typically, there are at least two types of scheduling occurring in an operating system: short-term and long-term. Long-term schedulers determine which possible requesters at a given time may actually request a resource. The short-term scheduler selects from among the active ‘‘requesters’’ that currently have need of the resource and allocates the resource to the selected ‘‘requester.’’ For example, if the requesters are processes and the resource is the CPU, the long-term scheduler manages the movement of processes from inactive (waiting in batch queues or in hibernation) to active (in wait or execute). The short-term scheduler, on the other hand, would determine which process should execute next on the CPU. Hybrid services between the two may also be available in the operating system.

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

The priority scheduling of resources allows the requester to have associated with it its importance to use the service. More complex schemes also have a criticalness of the request that is used for graceful degradation purposes. The scheduler(s) will use the priority information to arbitrate resource requests and to queue requests in the specific order. A priority scheduler may need to support multilevel queues to support proper execution.

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

4.2.4.1.2 Task Management Services

These services relate to the creation, management, and deletion of tasks executing within the scope of an operating system. These tasks are distinguished from ‘‘processes’’ via the following characteristics:

— There may be multiple threads of execution per address space.

— There is low overhead for context switches between threads located in the same address space.

In this context, ‘‘management’’ consists of those services that affect the execution of a task:

— Stop and restart execution of a task (e.g., suspend, resume).

— Modify processor allocation to a task (e.g., priority, timeslice).

— Modify scheduling of the task based on timer (or other) events.

— Protect the task from interruption during critical periods.

— Create a task and make it ready for execution.

— Destroy a task.

— Evaluate a reference to a task.

— Evaluate a connection to a task, where a connection is a logical communication path between any two tasks.

4.2.4.1.3 Environment Services

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

— Process-specific attributes (process identification, priority, stack size, scheduling attributes, status, memory allocation).

— Task-specific attributes (task identification, priority, scheduling attributes, status, memory allocation).

— Processor-specific attributes (node identification, electronic nameplate information).

— User-specific attributes (user identification and terminal ID, user interaction profile).

— Environment variables (command-line arguments, menu selections).

— Current time and date

4.2.4.1.4 Node Internal Communication and Synchronization Services

One or more applications and application subcomponents may run on a processor within an application platform simultaneously. The applications run as independent software entities and communicate among themselves via a variety of mechanisms provided or managed by the system services (see Figure 3-2). An important class of system services relates to the coordination and synchronization of these software entities. In traditional systems, entities execute on a single hardware processor. However, it is becoming common to have multiple processors and networked processors that place more requirements on the system services to provide coordination and synchronization among the many truly concurrent software entities.

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

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

— Create, delete, open, close, read, and write shared memory.

— Create, delete, read, and write event flags.

— Create, delete, set, and wait on semaphores.

— Create/send and receive signals.

— Create, delete, open, close, send to, get from, and control message queues.

— Create, delete, send, and receive streams.

4.2.4.1.5 Generalized Input/Output Services

These services are used by an application to perform generalized device I/O operations. These operations include synchronous and asynchronous operations for device and class specific functions. Specifically, these form the services needed to implement or include logical device drivers in a system. These services are device initialization, device attachment, asynchronous operation, and error notification. In addition, they include those services that are used to directly access specific device capabilities, particularly those services often referred to as ‘‘raw I/O.’’

4.2.4.1.6 File Oriented Services

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

Naming and Directory Services

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

File Modification Primitives

Primitive services for files and directories are:

— Read a portion of the file.

— Write to a portion of the file.

— Open access to a file.

— Create a new file.

— Close access to a file.

— Delete a file.

— Copy a file.

— Merge two or more files.

— Append one file to another.

— Split one file into two or more files.

— Support read and write locks at both the record and file levels.

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

File Support Services

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

Realtime Files

Realtime systems often need special files to ensure fast, bounded, and consistent performance in time critical situations. The need for a bounded response time for a given I/O function drives the design of these files and services. One service preallocates the complete disk space needed for a file at creation time, while another guarantees that records within files are aligned in an optimal way (such as along word boundaries). Services support the access of records within the file in ways that make response time constant or bounded, including by direct access.

4.2.4.1.7 Event, Error, and Exception Management Services

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

- Event and error receipt.
- Event and error distribution.
- Event and error management, including user-selectable error processing alternatives (filtering, retry, ignore, accumulate occurrences).
- Event logging.
- Enable/disable and mask/unmask interrupts.

4.2.4.1.8 Time Services

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

- Create a timer.
- Delete a timer.
- Initiate the measurement of an arbitrary specified time duration.
- Receive an indication when the specified duration has elapsed.
- Read the current value of a timer.

— Initialize a timer with a value and count direction (i.e., increment or decrement).

— Trigger a timer to begin incrementing or decrementing.

— Associate with a timer some action to be taken when the specified duration has elapsed.

4.2.4.1.9 Memory Management Services

These services are used by application processes and tasks to request additional memory and return it to the processor for reuse. They cover the services required to fulfill the needs of both virtual and fixed memory. Specifically, there is a service for locking pages in real memory to support the needs of virtual memory systems.

4.2.4.1.10 Logical Naming Services

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

The logical name to physical name relationship can be one to many, many to one, or many to many. Many times, one physical resource may have multiple logical names as well as one logical name representing a ''bank'' of available physical resources. These services must provide the proper resolution of names, logical and physical, in all of these cases.

4.2.4.1.11 System Initialization, Reinitialization, and Shutdown Services

System initialization consists of services for a complete restarting of the software, starting up the attached hardware subsystems devices, doing subsystem and system self tests, and completely initializing the database.

System reinitialization consists of services for restarting the software while using the existing database information. The software may have to be reloaded and the database may have been reestablished by a system recovery. Attached hardware subsystems may also need to be reinitialized.

Reinitialization also includes a function to restart applications redistributed to other processors after a processor module failure. Within a processor, there is a service to initialize applications in a system with the existing software, but with the database reinitialized. Also within a processor, there is a service to restart the applications in a system with the existing software and database retained.

Shutdown services are those required to perform planned orderly shutdown at the local and remote levels for each and all processor(s) throughout a system. These services support both crisis and non-crisis situations that call for system shutdown. They make sure that the persistent store is in a consistent state, see to the clean termination of all processes, programs, devices, etc., and take care of user notification. They also provide for the running of system diagnostics.

4.2.4.2 External Environment Interface Services

Data Interchange External Environment Interface Services are required by the System Services. Of particular interest are the formats, locations, and procedures for using system administration files, such as password files, system startup files, and configuration files.

4.2.4.3 Interapplication Software Entity Services

This could include support for generalized network/multisession services, such as message handling between system components, global object definition specification, and intermediate language definition.

4.2.4.4 Resource Management Services

These services provide general management functions across the entire platform. They consist primarily of system administration-oriented functions (i.e., management of system interfaces within the scope of the administrator, such as setting up defaults and limits.)

4.2.4.4.1 System Operator Services

The system operator needs to access and control the system services in order to allow the platform to perform properly. If a system has an operator, the major functions that need to be supported are system control, reconfiguration, and status reporting. Currently, these services are usually made available to an operator through a command language interpreter, which is an application program that accesses these system services.

Note that the Windowing Services provide the building blocks (menu utilities, command parsers, etc.) for building the user interface while the System Operator Services make available operating system status and control functions to appropriate application programs with the proper security level.

These services support general conventions and specifications for interaction between system components.

4.2.4.4.2 System Administration

These services and procedures are those required to assure management and allocation of system services to system users, both local and remote. They consist primarily of those services required to establish authorized users of the system, with associated allocation of processor resources, including memory, processor time, priority, and mass storage space. These services are both static (as in the establishment of a new user identification) and dynamic (as in login/logout).

4.2.5 Standards, Specifications, and Gaps

Table 4-2 – System Services Standards

Service	Type	Specification	Subclause	
Process Management	S	ISO/IEC 9945-1	4.2.5.1	E
Task Management	S	ISO/IEC 9945-1	4.2.5.1	E
Environment Services	S	ISO/IEC 9945-1	4.2.5.1	E
Node Internal Comm/Synch	S	ISO/IEC 9945-1	4.2.5.1	E
Generalized I/O	S	ISO/IEC 9945-1	4.2.5.1	E
	G	OSF AES – OSC	4.2.5.3	E
	G	SVID	4.2.5.3	E
File Oriented Services	S	ISO/IEC 9945-1	4.2.5.1	E
Event, Error, and Exception	S	ISO/IEC 9945-1	4.2.5.1	E
	G	OSF AES – OSC	4.2.5.3	E
	G	SVID	4.2.5.3	E
Time Services	S	ISO/IEC 9945-1	4.2.5.1	E
Memory Management	S	ISO/IEC 9945-1	4.2.5.1	E
Logical Naming	S	ISO/IEC 9945-1	4.2.5.1	E
System Init/Reinit/Shutdown	S	ISO/IEC 9945-1	4.2.5.1	E
	G	OSF AES – OSC	4.2.5.3	E
	G	SVID	4.2.5.3	E

4.2.5.1 Current Standards

Portable Operating System Interface (POSIX) Part 1

ISO/IEC 9945-1 (IEEE Std 1003.1) is the first in a set of planned international POSIX standards. It defines services and characteristics that need to be in the platform for portable applications, as do some of the other planned standards. Another type of POSIX-related standard is bindings for those services to specific languages. The third type deals with concepts that cross between various groupings of services, such as security and distributed processing.

The purpose of the ISO/IEC 9945-1 standard is to define a standard operating system interface based on the UNIX Operating System documentation to support application portability at the source level. The document is intended for systems implementors and applications software developers.

In addition to ISO/IEC 9945-1, ISO is planning to publish another standard (as yet unnumbered) on test methods for verification of POSIX standards, which will be identical to IEEE Std 1003.3-1991.

Table 4-3 outlines the contents of POSIX.1 {2}. This document is identical in its ISO/IEC form (ISO/IEC 9945-1) and the US national standard form (IEEE Std 1003.1). Revisions are currently in progress to deal with:

- A language-independent services specification
- A unified data interchange format
- Service interfaces for control of character cell terminals
- Miscellaneous functions identified in comments on the current standard.

Table 4-3 – Functionality of POSIX.1 Standard

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

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

NOTE: FIPS 151-1 is a profile of the base standard POSIX.1 {2}.

4.2.5.2 Emerging Standards

IEEE P1003.4

The IEEE P1003.4 Group is defining realtime extensions to ISO/IEC 9945-1. Draft 9 of the realtime POSIX extensions proposes standardized interfaces to the following functions:

- Response to asynchronous events
- Priority interrupts and scheduling
- Preemptive scheduling
- Memory locking
- High-performance file system (contiguous or other)
- Realtime timers (with nanosecond resolution times)
- Shared memory
- Semaphores
- Interprocess communications (message passing)
- Asynchronous event notification
- Synchronous input and output.

The P1003.4 group is also specifying an interface to threads (P1003.4a).

4.2.5.3 Gaps in Available Standards

While ISO/IEC 9945-1 and P1003.4 both represent very important work, they do not yet address all of the services indicated in 4.2.4. Areas of particular shortfall include Event, Error, and Exception Management Services, some Generalized I/O Services (particularly concerning services for device drivers), and System Initialization, Reinitialization, and Shutdown Services. In addition, Security (see 5.2) and Reliability, Adaptability, and Maintainability services are not reflected in these two base standards, and some capabilities are explicitly considered to be implementation defined. For some of the services discussed here, adequate consideration is not given to the implications of multiprocessor and distributed implementations of the services and interface provided. Finally, since these are intended to be base standards (or, in the case of P1003.4, an extension to a base standard), profiles are needed in order to select appropriate features and provide appropriate combinations with other related capabilities.

4.2.5.3.1 Public Specifications E

The following are public specifications that define interfaces to services for which no formal standards are currently available.

OSF /1

The Open Software Foundation (OSF) ''Application Environment Specification (AES)—Operating System Component'' (OSC).

Service Gaps Addressed:

- — Generalized I/O
- — Event, Error, and Exception
- — System Init/Reinit/Shutdown

SVID

The AT&T System V Interface Definition (SVID), Issue 3.

Service Gaps Addressed:

- — Generalized I/O
- — Event, Error, and Exception
- — System Init/Reinit/Shutdown

XPG 3 E

X/Open's XPG3 specifications. E

Service Gaps Addressed: E

- — Generalized I/O E
- — Event, Error, and Exception E
- — System Init/Reinit/Shutdown E

4.2.5.3.2 Unsatisfied Service Requirements

There are two significant areas of the services described above for which no standards currently exist. One is the considerations implied by the use of multiprocessors to implement some or all of the services described herein. The other area is that of interfaces to logical device drivers.

4.2.6 OSE Cross-Category Services

4.2.6.1 Capability and Security Services

These services support the ability of the system to control usage such that system integrity is protected from inadvertent or malicious misuse. These protection services provide a mechanism for the enforcement of the policies governing resource usage. Note that many of the security services are implicit services; i.e., they are provided without an explicit request to the operating system. There are two distinct classes of system access with which operating system services must be concerned: physical access and logical access.

Security services at the physical level are used to protect against security compromise, given unauthorized personnel may have physical access to system hardware. Typically, the physical access is to a terminal and/or terminal/display cables; however, physical access may also include network cables, central processing units, disk drives, or tape drives. Prevention of physical access by unauthorized personnel may require different operating system services under different circumstances.

Logical access is the ability to interact with the operating system via a terminal/display. Security services at the logical level can be implemented through passwords and watchdog timers.

Capability services attach operation lists that limit a process's ability to act on resource objects. This is to ensure the resources are not misused. Access to resources can be protected by services using capability lists as well as access lists, lock/key mechanisms, global tables, or through dynamic protection structure services.

Prevention of Unauthorized Access

The system may need to be guarded from attempted access by unauthorized personnel. The point of access to the operating system that is typically of concern is through the API. Given the mode of operation (system high, multilevel, open) at which the system is operating, these services differ and have differing implications on other system services (such as reliability and naming) and system performance.

Prevention of Data Compromise

These services prevent access of data by users not authorized to the data. These services may be implemented using access lists on files (and directories) and/or encryption of data or in other ways.

Prevention of Service Denial

These services ensure that a service request will be met by the operating system in a reasonable time if the requester is authorized to use the service. These services ensure that a bandit user or process cannot cause system malfunction by monopolizing system services or resources.

Security Administration

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

E

4.2.7 Related Standards

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

P1003.6	Security Interface for POSIX.
P1003.12	Protocol Independent Interfaces (for networks).
P1238	OSI Application Program Interfaces (initial effort is to provide at least sufficient facilities for the support of FTAM API specifications).

E

4.3 Network Services

Responsibility: Charles Severance

4.3.1 Overview and Rationale

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

Applications gain direct access to network services via the POSIX API. The network is just another system resource (albeit an important one) allocated among the competing processes.

4.3.2 Scope

Network services cover the areas of file transfer, namespace and directory services, electronic mail services, services in support of distributed environments such as remote procedure call, distributed time management, transparent file access, and data representation services. The application programs using these services should be able to access them via a high-level, context-insensitive or low-level, context-dependent interface.

In the open systems and distributed system environments, interoperability is of equal or greater importance than portability. The network protocols defined for both Open Systems Interconnect (OSI) and Internet Protocol Suite (IPS) for TCP/IP should provide the basis for the open networking interfaces; however, these interfaces should not preclude the use of some subsequent networking protocol in the future. The interfaces provided by the network services must be network protocol independent and provide for this level of interoperability.

It is important for an open system to interoperate with more systems than just other open systems. Many open systems users will have requirements to interoperate with non-OSI networks for the near future.

4.3.3 Reference Model

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

As illustrated in Figure 4-3, the components of a network architecture that require standardization are divided into two groups called external environment interfaces (EEI) and application program interfaces (API).

There may be some correspondence between services offered to the application across the API and the interfaces available at the EEI. It is quite possible for an API service to have no corresponding effect at the EEI. A good example of this is an interapplication communication service provided by the Network API between

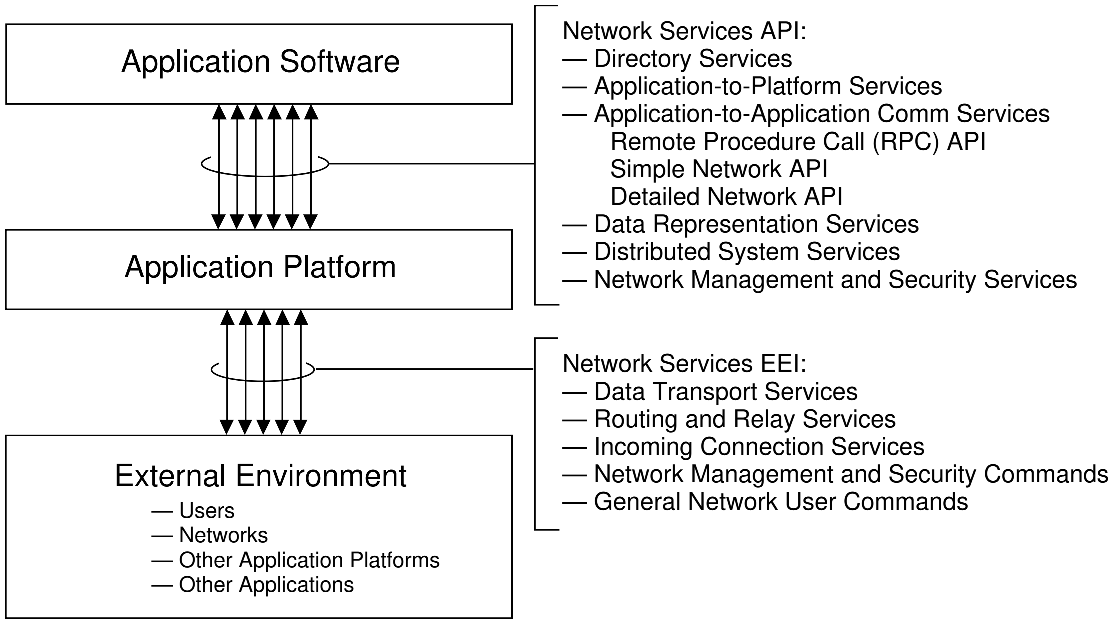

Figure 4-3 – POSIX Networking Reference Model

two applications on the same application platform. There may also be services available at the EEI provided by the Application Platform that are not available at the API such as remote login services.

4.3.3.1 Network Application Program Interface (API) Services

The API is concerned with the interfaces and associated standards that apply to the interface between the application and the application platform.

The services available at the API are:

- — Directory Services
- — Application to Platform Services
- — Application to Application Communication Services
- — Data Representation Services Services
- — Distributed System Services
- — Network Management and Security Services

Directory Services are those services associated with identifying and naming network elements.

Application to Platform Services provide an application with a very high level interface to networking capabilities. This interface provides applications with capabilities such as ‘‘mail this file to this address’’ or ‘‘transfer user xxx file from host yyy to the local host.'' These services do not require the application to be aware of any of the low level network details.

Application to Application Services are the services provided by the Application Platform that allow an application to communicate with another application to exchange information. These interfaces support applications that range from having extremely simple networking requirements to the most complicated applications that must make full use of every possible network capability.

Data Representation Services provide the application with network oriented data representation services to insure the application can interchange information with other entities in the proper format.

Distributed system services provide the application with the ability to make use of multiple physical computer systems resources.

Network management and security services allow the application to control and configure the network resources.

4.3.3.2 External Environment Interface Elements

4.3.3.2.1 User Interface EEI Elements

The User interface EEI elements include the commands that users can use to perform network functions such as:

— File transfer

— Electronic mail

— Remote printing

These commands are considered to be beyond the scope of this clause and will be covered in 4.10.

The User interface EEI elements that will be covered in this section are the commands that are used to perform network management and security functions.

4.3.3.2.2 Communication EEI Elements

The primary focus of the network EEI is the network protocols and supporting formats for network communication.

The entities in the external environment may be other application platforms or user interface equipment connected to the network using the open networking protocols. The standards at the EEI will be in several areas including:

— Physical connections

— Network protocols and formats

— Distributed systems services

The standards at the EEI will impact system interoperability but also may have an effect on application portability because certain applications may require particular types of network access to operate.

4.3.3.3 Implementation Aspects

The POSIX OSE Network reference model focuses on the requirements of application portability and system interoperability. As such, the model does not represent how systems are actually put together.

In the network area, there is much effort dedicated to the design of network standards to allow network components to be re-usable. This subclause shows how some of these network standards are related within the POSIX Network Reference Model.

Other network models are also related to the POSIX OSE Network Reference models. None of these other models are in conflict with the POSIX OSE Network Reference model. These models show much more detail in the area of how different standards work together.

4.3.3.3.1 Relationship Between the OSI Reference Model and the POSIX OSE Network Reference Model

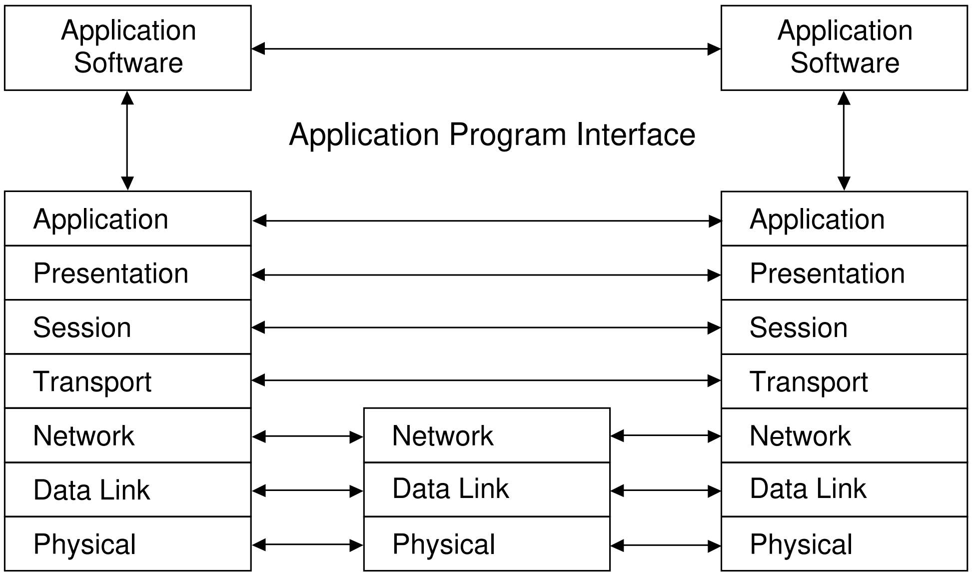

Figure 4-4 – OSI Reference Model

Figure 4-4 shows the OSI reference model for networking as standardized by ISO. E

There are many aspects of network architecture that are specified by the OSI E
reference model: E

- The number of layers in the model and the roles for each layer.
- An indication of which layers are logically end to end and which layers are simply to the next physical network node.

— The services between the layers and the protocols between the peers within the same layer. This has an impact on the actual format of the information transferred between nodes at the physical layer.

In addition, this model specifies how networks of computer systems can be assembled using the routing capabilities of intermediate nodes.

The POSIX OSE Network Reference Model has a much more limited scope than the OSI reference model. The POSIX OSE reference model only looks at two interfaces to an application platform: the interface between application software and the application platform (API) and the interface between the application Platform and the External Environment (EEI). At both the API and EEI, the POSIX OSE network model describes the services that are provided to the application or external environment at the interface.

Figure 4-5 shows an example of how an application platform made up of a single computer system would provide services at the API and EEI. It is important to note that the POSIX OSE application platform actually may be made up of multiple physical computer systems, as shown in Figure 3-5. In Figure 3-5, each computer system making up the distributed system would be running a complete OSI stack for networking.

Because the OSI portions of the Application Platform External Environment Interface depend on the format, protocol, and services of what is produced at the physical level of the OSI reference model, the EEI technically depends on all seven layers the OSI model plus the services added on top of the application layer such as platform provided services or network management services.

Figure 4-6 shows an API interface to only layer seven of the OSI Network interface, which is intended to be the primary API for accessing network services. It is possible to define APIs that interact directly with any of the seven layers. There are a number of pragmatic reasons to provide APIs that access layers below layer 7. The cost of using one of these lower layer APIs is that the applications may sacrifice portability and/or interoperability.

It is important to note that while these APIs are represented as a part of a layered network architecture, from the point of view of the application interacting with the application platform, this layering is not critical to the use of the services. From the application perspective, there are simply three different types of network services, each with a different set of capabilities and requirements. Whether or not there is any actual layering or code common to the three services is implementation dependent.

4.3.3.3.2 POSIX Network Standards Efforts

The current POSIX approach to networking focuses on producing Application Program Interface (API) specifications. Most of the network connectivity specifications at the External Environment Interface are well covered on other standardization areas such as ISO (OSI networking) and the MIL-STD process (TCP/IP).

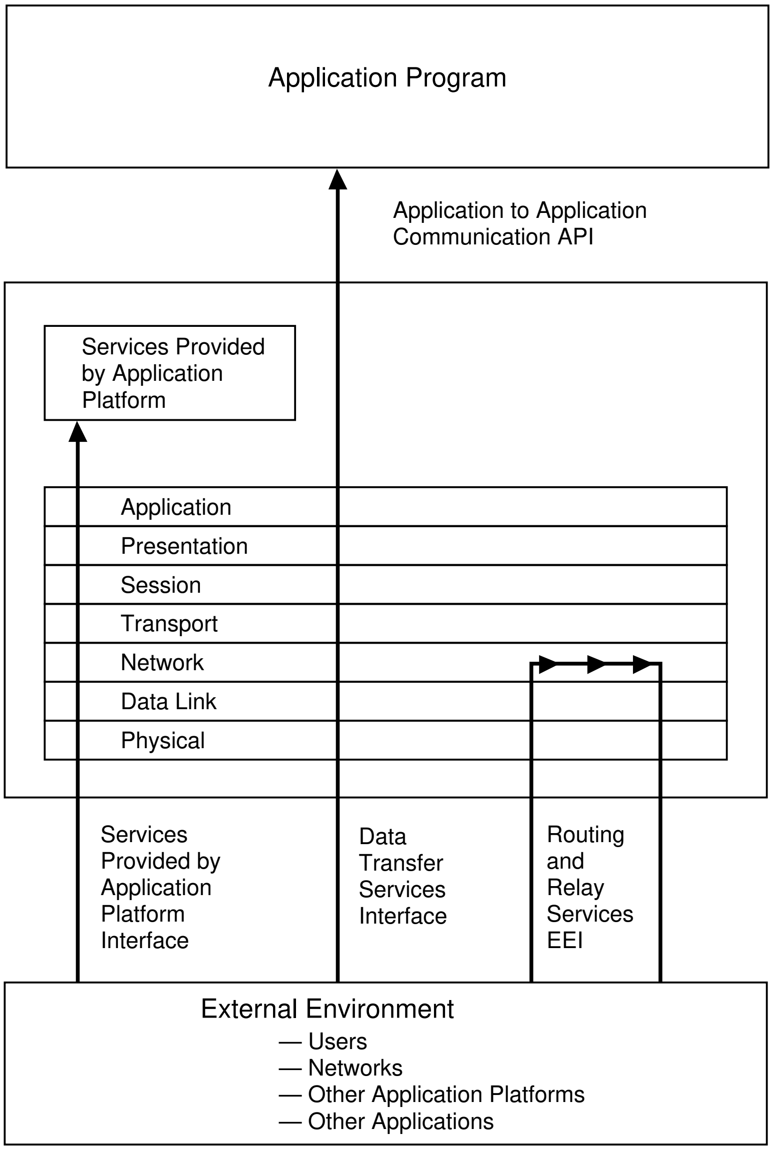

Figure 4-5 – Relationship of OSI and POSIX OSE Network Reference Models

One important aspect of the POSIX networking approach is that it is not focusing solely on producing standard APIs for OSI Network services. The POSIX Simple Network Interface (P1003.12 SNI) is explicitly designed so to be implemented transparently on a wide variety of networks. At the current time the possible list includes:

- OSI Application Layer
- OSI Transport Layer

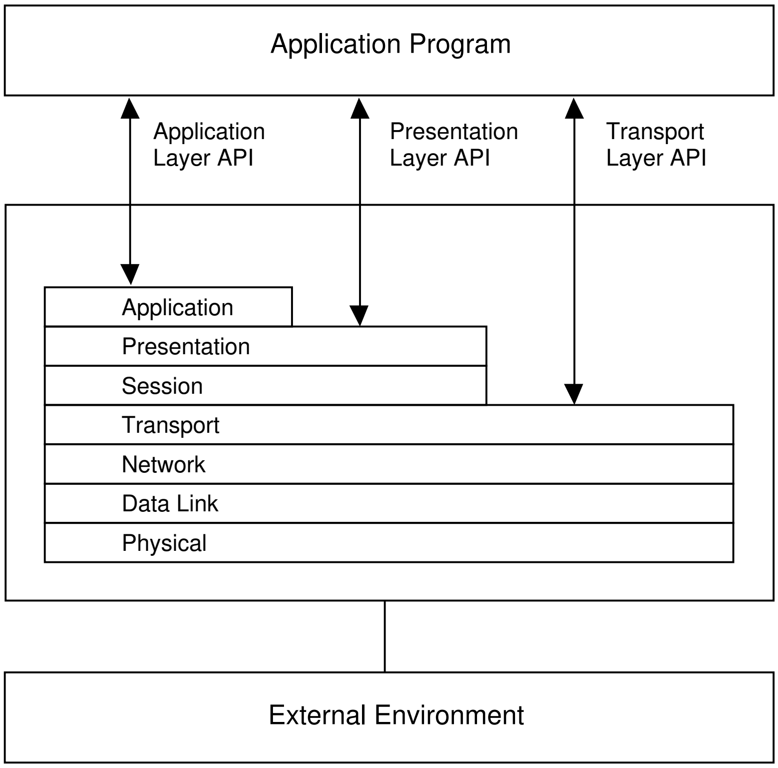

Figure 4-6 – Multiple POSIX OSE APIs to Different OSI Layers

— Internet Protocol Suite (IPS) E

— Other networks, including proprietary networks

The current POSIX API standardization efforts include:

P1003.12	Simple Network API	
P1003.12	Detailed Network API	
P1003.17	Directory Services API	
P1224	X.400 Electronic Mail Services API	E
P1224.1	OSI Object Management API	E
P1238.0	OSI Application Layer API (ASCE)	
P1238.1	OSI Application Layer API (FTAM)	

Figure 4-7 shows how the basic network services can be related. The Simple Network Services API is designed so that a Simple Network Services Implementation can be done using the services available using the Detailed Network Interface API. An application can use the Detailed Network Interface to access multiple network transports but there may be differences between networks visible at the

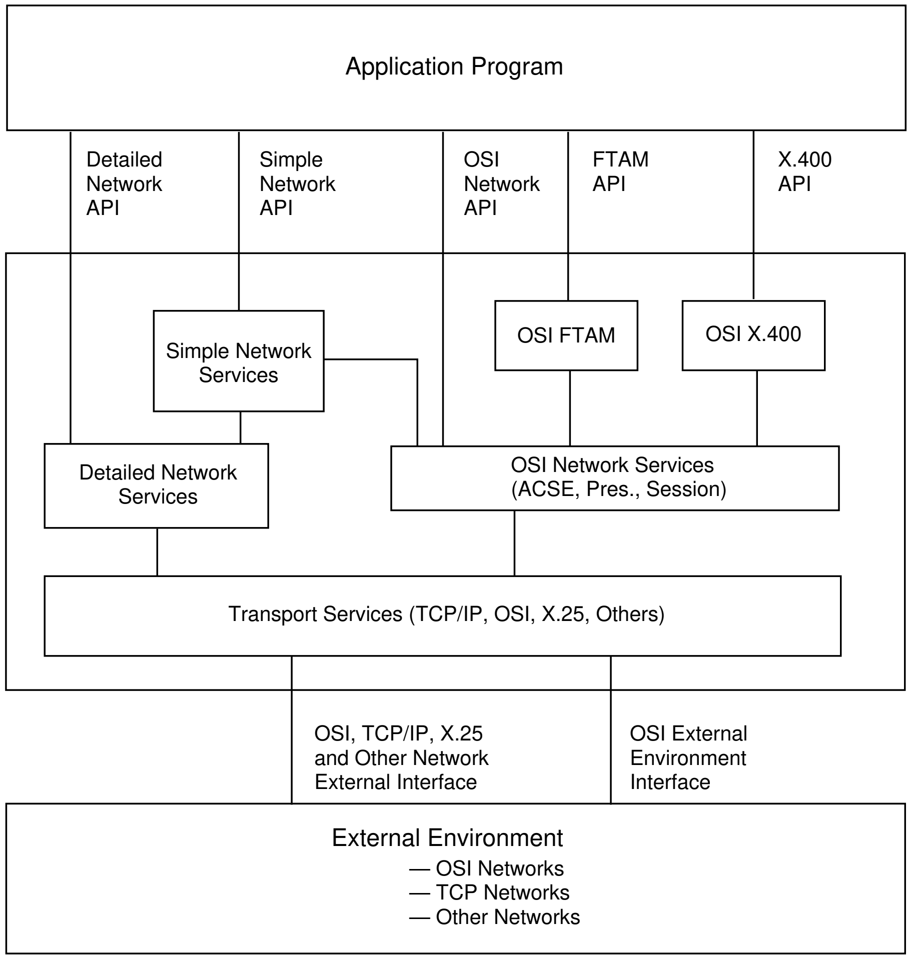

Figure 4-7 – POSIX Network Services Model E

API. Applications that need to be portable across different types of network transports should be written using the Simple Networking Interface.

It is important to note that while the SNI API and DNI API standards have been designed so that the SNI Services can make use of the DNI API to access transport services, it is not a requirement that every implementation of SNI Services be written using the DNI API to access transport services. From the point of view of the application program, it is only important that the application platform provide an API for both the SNI and DNI services. This interface between the SNI Services and the Transport Services is an example of a Systems Internal Interface, as described in 3.6.

Another example of a System Internal Interface that is the subject of discussion in the POSIX Network area is the interface between the OSI Network Services and the transport services. This may or may not be required to be the DNI API. This is an example of an interface that should have no impact on user application portability but may have great impact on the ability to procure the different types of network services from different vendors.

The area of Directory Services (P1003.17) is also specified so to be able to make use of different types of Directory Services including:

— X.500 Directory Services

— TCP/IP Directory Services

— IEEE P1003.7 System Administration and Management Services

Figure 4-8 shows how the Directory Services are related to the other network services. All of the APIs and SIIs from the previous figure have been eliminated to reduce the number of interfaces shown on the figure.

4.3.4 Service Requirements

The service requirements for the network component of an open system are very wide ranging. Many of the other components of the application platform make implicit or explicit use of network services.

Much standardization effort has gone into the aspects of networking that are available at the external environment interface. Effective networking standards at the external interface are fundamental to providing system interoperability.

The service requirements for both the API and EEI are described in this section.

4.3.4.1 Application Program Interface Services

4.3.4.1.1 Directory Services

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

— Look up the name to be used to access a particular service

— Look up the address of a named object

4.3.4.1.2 Application to System Services

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

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

(1) File transfer

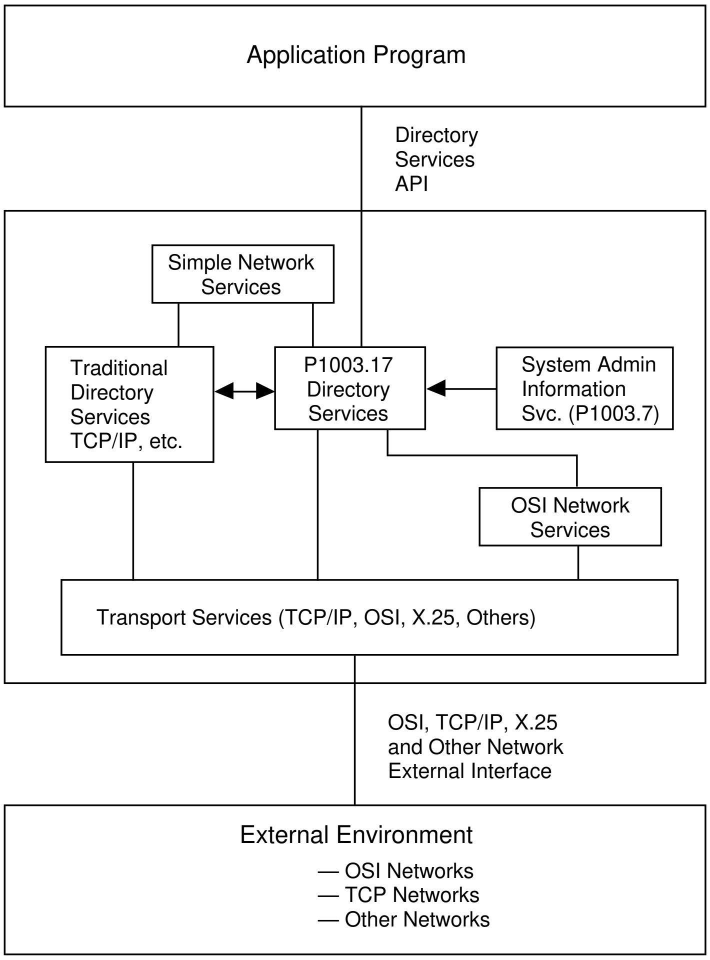

Figure 4-8 – Directory Services Architecture

(2) Remote execution of commands

(3) Electronic mail E

(4) Remote login

(5) Remote printer access

(6) Network status

— The ability to access remote or local systems using remote procedure calls (RPC). When this type of access is provided, nearly all of the details of the network connection and interaction are masked from the application.

4.3.4.1.3 Application to Application Service

There are three areas of application to application service requirements:

- RPC Services
- Simple Network Services
- Detailed Network Services

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

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

The Simple Network Services are application to application services provided using a simple set of interface routines. These will allow a wide variety of networking applications to be written that do not need to exercise control their network access at a very complex level of detail.

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

Applications written using the simple network services may not be able to make use of unique advantages of a particular physical networking scheme. To make use of these network-specific features the Detailed Network Services must be used.

The service requirements for the simple network services are intended to be the minimum requirements to write a large subset of network applications.

The Simple Network Services sacrifice the capability to control every detail of the network services in the interest of portability across networking environments and applications simplicity.

The Detailed Network Services API allows the application to control over much more detail of the network services. In addition, using the Detailed Network Services an application may be able to make use of unique networking capabilities available in particular networking environments.

4.3.4.1.3.1 RPC Services

These service requirements include the ability:

- To register as an RPC service provider
- To wait for incoming requests
- For an application using RPC services to control parameters such as timeout

4.3.4.1.3.2 Simple Network Services

The services provided at the simple network interface are:

(1) Name resolution

(2) Connection oriented services

- The ability to indicate willingness to accept incoming connections
- Establishing and destroying connections
- Data transfer over connections
  - Read
  - Read with timeout
  - Write
  - Write with timeout
- Simple error handling
  - Connection dropped notification
  - Connection read failure
  - Connection write failure
- The ability to close a connection
  - Unconditionally
  - Only after all data has been received

(3) Connectionless services

- The ability to indicate willingness to accept incoming requests
- The ability to send requests
  - With acknowledgment
  - Without acknowledgment
  - Specified timeout
- The ability to receive requests
  - Wait unconditionally
  - Wait with timeout
- The ability to query as to whether any requests are available
- Simple event notification
  - Lost request
  - Request acknowledgment

- Simple error handling
  - General network failure

(4) Support for server applications

- The ability to register as the provider for a service

(5) Simple status inquiry

- General network availability

4.3.4.1.3.3 Detailed Network Service Requirements

The services provided at the Detailed Networking Interface include all of the service requirements in the Simple Network Service Requirements plus the following abilities:

(1) Query the network services to get detailed information about network configuration and status

(2) Specify performance metrics

(3) Control routing

(4) Select between different network protocols

(5) Negotiate capabilities

- Required capabilities
- Optional capabilities
- Determine the results of the negotiation

(6) Information with different priorities

(7) Request and process extended event notification

(8) Request and process extended error recovery including allowing the application to completely control error recovery.

(9) Make full use of network resources for performance critical applications

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

4.3.4.1.4 Data Representation Services

- The ability to access all of the data representation and format conversion services to allow an application to communicate with a wide variety of computer systems.

4.3.4.1.5 Distributed System Services

The services provided in this area include the ability to:

— Identify available resources in a distributed system

— Dynamically make use of the resources in a distributed system.

— Access files regardless of the physical location of the files.

— Have reliable time services across all of the resources of the distributed system.

4.3.4.1.6 Network Management Services

The services provided at the Network Management API are abilities to:

(1) Manage

- — Network objects
- — Network relationships
- — Network security

(2) Monitor and report on

- — Network events
- — Network service alarms
- — Network security alarms

(3) Log

- — Network events
- — Network availability
- — Network load
- — Network performance

(4) Test network performance and reliability

4.3.4.2 External Environment Interface Services

At the external interface, there are several types of services that are provided. These include:

— Data transfer and connectivity

— Routing and relay services

— Services provided by the application platform directly to an incoming connection

— Network management and security services provided to other networks and other nodes within a network

— Network management user interface

This clause does not address the user interface to the general network services such as file transfer or mail sending. That is covered by the command interface clause, 4.10. As stated above, this clause covers the network management user interface.

In addition, there are a number of other areas of external interface requirements that are not covered in this guide. They include:

— Physical network interface connections

— Electrical specifications for network connections

— Specifications for physical network construction

4.3.4.2.1 Data Transfer and Connectivity

Services required at the EEI in the area of data transfer and connectivity include the ability to:

— Connect and interoperate with other standards-based systems using standards-based protocols including X.25 and OSI.

— Connect and interoperate with systems using de facto networking standards such as TCP/IP and UUCP.

— Connect and interoperate with personal computer and workstation networks.

— Interoperate with industry leading networking interfaces.

4.3.4.2.2 Routing and Relay Services

Services required at the EEI in the area of routing and relay capabilities include the ability to:

(1) Relay information through a system between like networks.

(2) Gateway information through a system between unlike networks at a data transfer level. Examples of this type of gateway include:

   — Local Area Network (LAN) to LAN

   — LAN to Wide Area Network (WAN)

   — WAN to Global Area Network (GAN)

   — Networks to point-to-point connections

   — Point-to-point connections to networks

(3) Convert information from one format to another when transferring between unlike computer systems or networks. Information that may need to be converted includes:

   — Mail messages

   — File contents

— Printer file contents

The primary requirement for the routing and gateway services is to make any necessary relays and gateways transparent to the applications and systems using the network. This requires complete gateways and relays.

4.3.4.2.3 Services Provided by the Application Platform at the EEI

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

— Remote logon and terminal emulation

— Remote execution of commands

— File transfer services

— Remote authentication

— Remote data access

— Remote status information

— Mail delivery services

— Directory services

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

4.3.5 Standards, Specifications, and Gaps

4.3.5.1 Current Standards

Table 4-4 lists standards that address the services outlined in this clause. This table includes international standards, emerging standards coming from national and international bodies, and other current standards that meet gaps in the service requirements. Public specifications are cited to fill gaps only when there are no existing or emerging standards to meet the service requirements.

ISO Protocol Stack Standards

Figure 4-9 describes how the ISO protocol standards cited in this guide fit together.

4.3.5.2 Emerging Standards

Table 4-4 – Networking Standards

Service	Type	Specification	Subclause	E
Directory Services	S	X.500	4.3.5.1	E
	E	IEEE P1003.17 X.500 API	4.3.5.2	E
Message Handling	S	ISO 10021 X.400	4.3.5.1	E
	E	IEEE P1224 X.400 API	4.3.5.2	E
File Transfer	S	ISO 857, ISO 8613, ISO 10026, ISO 8650, ISO 8652, ISO 8653, ISO 9735, ISO 9594	4.3.5.1	E
	E	IEEE P1238 FTAM API	4.3.5.2	E
Print Services	E	X3H3	4.3.5.2	E
Application Services				E
Connectionless	S	ISO 8649-2, ISO 8650-1	4.3.5.1	E
Connection Oriented	S	ISO 10040, ISO 10164, ISO 10165, ISO 9595, ISO 9596, ISO 9579	4.3.5.1	E
	E	IEEE P1238.1 ASCE API	4.3.5.2	E
Data Representation	S	ISO 8823 Presentation Protocol	4.3.5.1	E
	S	ISO 9576, ISO 8824, ISO 8825 ASN.1	4.3.5.1	E
Protocols				E
Session	S	ISO 8327, ISO 9548	4.3.5.1	E
Transport	S	CCITT X.214, X.224 (TP0)	4.3.5.1	E
	S	ISO 8072, ISO 8602 (TP4)	4.3.5.1	E
	E	IEEE P1003.12 Transport API ??	4.3.5.2	E
Network	S	CCITT X.25 PLP, ISO 8208	4.3.5.1	E
	S	ISO 8348 AD1, ISO 8473	4.3.5.1	E
Data Link	S	ISO 7776 HDLC/LAPB	4.3.5.1	E
	S	ISO 8802-2 Logical Link Control	4.3.5.1	E
Physical	S	EIA RS-232	4.3.5.1	E
	G	MIL-STD-114A	4.3.5.3	E
	S	ISO 8802-3 (CSMA/CD) ISO 8802-4 (Token Bus), ISO 8802-5 (Token Ring)	4.3.5.1	E

Table 4-4 – Networking Standards (*concluded*)

Service	Type	Specification	Subclause	
Network Management	S	ISO 9596	4.3.5.1	E
	S	ISO 9593	4.3.5.1	E
	S	ISO/NMF	4.3.5.1	E
Network Security	S	ISO 803.10	4.3.5.1	E
	E	X3T4	4.3.5.2	E
	E	SIRS 233	4.3.5.2	E
Distributed System Services	S	ISO DP	4.3.5.1	E
	E	IEEE P1003.8 TFA API	4.3.5.2	E
Remote Procedure Call (RPC)	E	ECMA 127	4.3.5.2	E
	E	ISO 10148	4.3.5.2	E
	E	IEEE P1237 API	4.3.5.2	E
Protocol-Independent Network Interface	E	IEEE P1003.12 SNI API	4.3.5.2	E
Interoperable Networking Directory Services	G	RFC-1034 Domain Naming	4.3.5.3	E
	E	IEEE P1003.17 Directory Services API	4.3.5.2	E
File Transfer	G	MIL-STD-1780 (TCP/IP FTP)	4.3.5.3	E
Message Handling	G	MIL-STD-1781 (TCP/IP SMTP)	4.3.5.3	E
Virtual Terminal	G	MIL-STD-1782 (TCP/IP Telnet)	4.3.5.3	E
Protocols	G	MIL-STD-1777 (IP)	4.3.5.3	E
	G	MIL-STD-1778 (TCP)	4.3.5.3	E
	E	IEEE P1003.12 API	4.3.5.2	E
Mainframe Networking	E	IEEE P1003.12 API	4.3.5.2	E
	G	X/Open CPIC	4.3.5.3	E
PC Networking	G	X/Open PCI:SMB	4.3.5.3	E

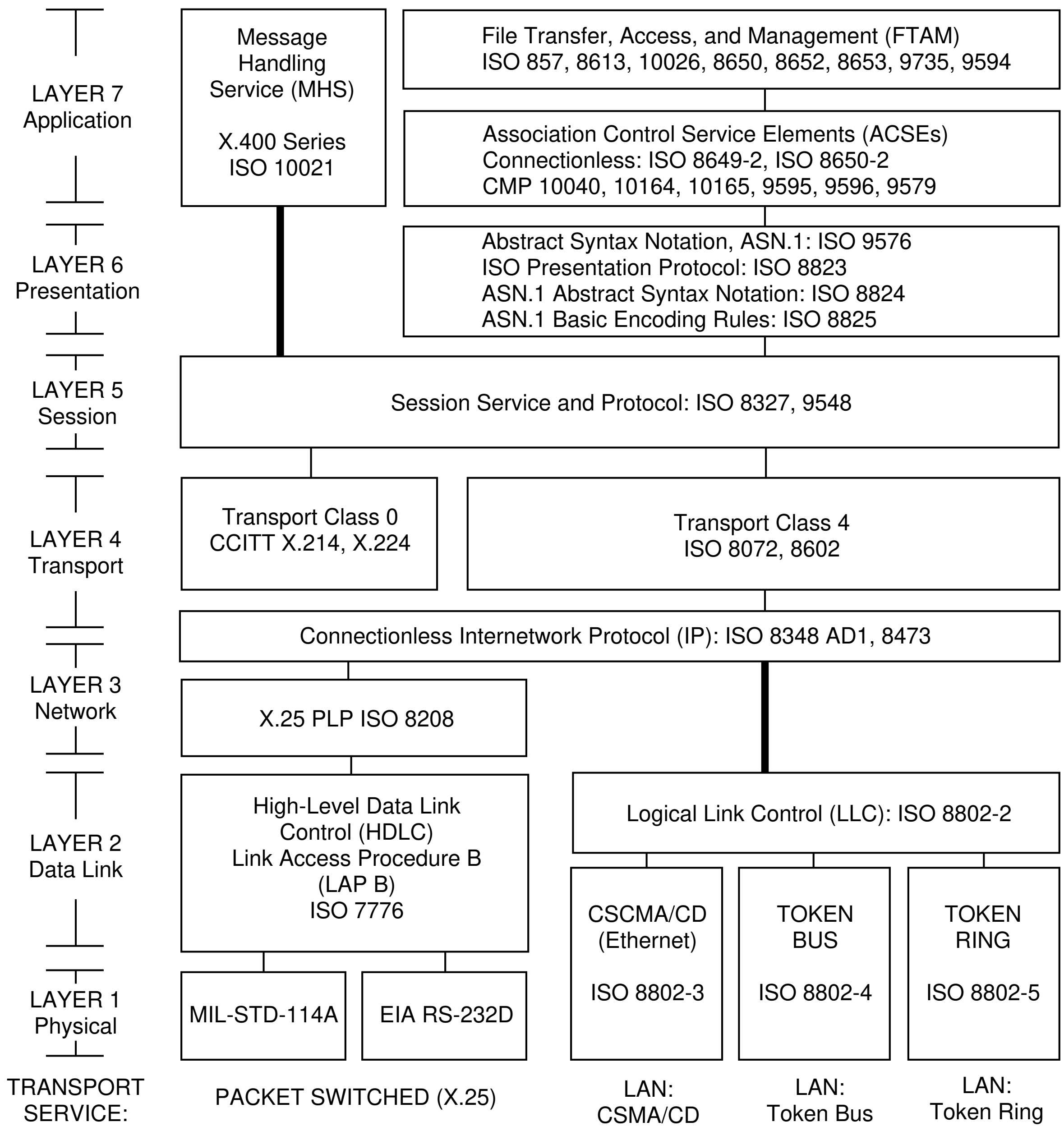

Figure 4-9 – OSI Network Services Standards

IEEE P1003.12

This group is developing a standard that provides networking application program interfaces. P1003.12 contains the specification for a Simple Network Interface (SNI) and a Detailed Network Interface (DNI). The Simple Network Interface is designed to usable on a number of different transport services, ranging from ISO networks to completely proprietary networks, without requiring application changes. To do this, the SNI has a very limited set of services with minimal parameters. The Detailed Network Interface is also designed to be implementable across a wide variety of network protocols. However, DNI allows applications to access the low-level details of each of the different network protocols. As a result, programs written using DNI may be portable between environments that use the same underlying network protocols.

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

IEEE P1003.17

This group is developing an API standard that will enable applications to access directory services. Backwards compatibility with existing name resolution services, such as TCP/IP, is included in the design of the P1003.17 interface. P1003.17 can also use the following directory services:

- X.500
- TCP/IP
- IEEE P1003.17 System Management Name Space
- Others

IEEE P1238

This group is developing an API for connection-oriented Application Layer services. It establishes a specification methodology and defines an API to:

- OSI Association Control Service Element (ACSE) services and
- common support functions for OSI connection-oriented protocol APIs.

The specification is operating system and language neutral; POSIX and C-language bindings are provided. Further, it is intended to be used as the basis for the connection management interface for the future Application Service Elements (ASE) such as the File Transfer, Access, and Management (FTAM) API.

IEEE P1238.1

This group is developing an API for interfacing with the FTAM application layer element. It is standardizing an X.400 API and a companion OSI Object Management API, based on the X.400 API and an OSI Management API developed by the X.400 API Association and X/Open. The X.400 API consists of two parts: an X.400 application API and an X.400 gateway API. These APIs were developed based on the 1988 CCITT X.400 series of recommendations. The X.400 API and Object Management API are separate documents.

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

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

4.3.5.3 Gaps in Available Standards

This subclause describes the standards that are cited to satisfy identified service requirements that are not satisfied by any existing or emerging standard.

Interoperable Networking Standards

This set of protocol standards is the traditional TCP/IP suite of standards, which are currently widely available on many computer platforms and operating systems.

This group of specifications includes:

TCP/IP	MIL-STD-1777, MIL-STD-1778
TELNET	MIL-STD-1782
FTP	MIL-STD-1780
SMTP	MIL-STD-1781

While these protocols are not expected to be standardized at any higher level than the MIL-STD level shown, it will be necessary for open systems to interoperate with these standards in a backwards-compatibility mode for some time.

Low Cost Wide Area Networking

The UUCP (UNIX-to-UNIX Copy Protocol) services and commands, for electronic mail and file copying, which are traditionally included in UNIX and UNIX-like systems are not addressed by any standards effort. Among other reasons, UUCP is not currently being addressed because of the inability of the POSIX groups to decide whether the UUCP services and commands should be standardized in the POSIX.2 Group (since UUCP is a traditional UNIX service with traditional command interfaces) or in the networking groups (since UUCP is an electronic mail and file copying facility that works on networks).

4.3.6 OSE Cross-Category Services

These EEI Services allow remote systems to be managed and monitored. Network management services include the ability to:

- Get network status information
- Get network configuration information
- Test network functionality
- Make network configuration changes

The security services allow the system management to control access to system resources and system information. Security services include:

- Protect the system from intruders
- Provide selective access to sensitive system resources
- Manage the network security

See also 5.3.

4.3.7 Related Standards

ISO 8587, Distributed Transaction Services; see 4.6.

4.4 Database Services

Responsibility: Sandra Swearingen

4.4.1 Overview and Rationale

This subclause describes an overview of an architectural framework for discussing database management. It also describes the services provided to application programs and users, and it describes standards, current and emerging, that standardize those database services.

Database management is an important component of the POSIX Open System Environment; in a large class of application programs, especially those used in business, database access through a database management system plays a key role. For portability and interoperability, an application using a database must be isolated from the hardware and software retrieval methods as much as possible. Otherwise the application must have the data manipulation capability coded in its own programs. This might be done if performance is a key issue and the data is very unique. The cost is portability and interoperability.

E

4.4.2 Scope

Included within the component of database management are various structured ‘‘data models,’’ including indexed files and network, relational, semantic, and object-oriented databases. Specifically excluded from consideration are services for accessing data that is not part of a database. This subclause discusses database management services from both the application program and user points of view.

Database services provided to application programs by this component, for example, include the ability to create, alter, or drop tables, records, and fields and the ability to insert, select, and update data included in the structures above.

Included within this component are also utility capabilities such as database administration services.

E

4.4.3 Reference Model

4.4.3.1 Reference Model

In this subclause, the conventional view of Database Management is related to the POSIX reference model described earlier.

The application programmer’s view of the database model is introduced first. Quite simply, an application program, through a *Database API*, requests database services. For convenience in the following discussion, the agent responsible for providing those services is called the *Database Manager* . The database manager is responsible for providing the application access to the *Database* . See Figure 4-10.

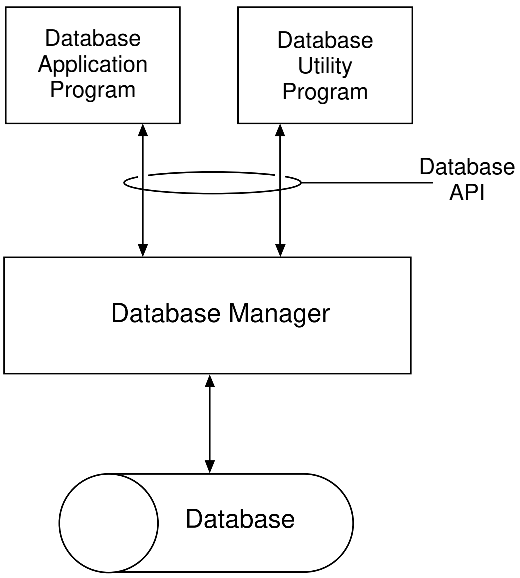

Figure 4-10 – The Traditional Database Model

This figure also demonstrates the concept of a *Database Utility Program* : one or more special application programs, usually provided by a database vendor, that perform utility services on the database. Such utilities might reorganize the database, recover the database after a system failure, etc.

The traditional database model can be incorporated into the POSIX reference model, as in Figure 4-11. This depiction of the model shows that the database manager is really just part of the overall POSIX Open System Environment and is available to the application through the POSIX OSE API.

The model depicted in Figure 4-11. is sufficient to describe an application developer's view of the POSIX OSE API in general, and the database API specifically. The four lines labeled ''Database API''represent the Database Applications Program Interface services, which are discussed in 4.4.4.1.

4.4.3.2 Implementation Aspects

Some real world considerations of the POSIX Reference Model were discussed in 3.6. One of the real-world approaches described is ''layering.'' Note that in the marketplace, Database Managers are often independently purchasable components that are effectively implemented as layers. Another consideration is Database Manager portability. It is not a requirement that a a database manager

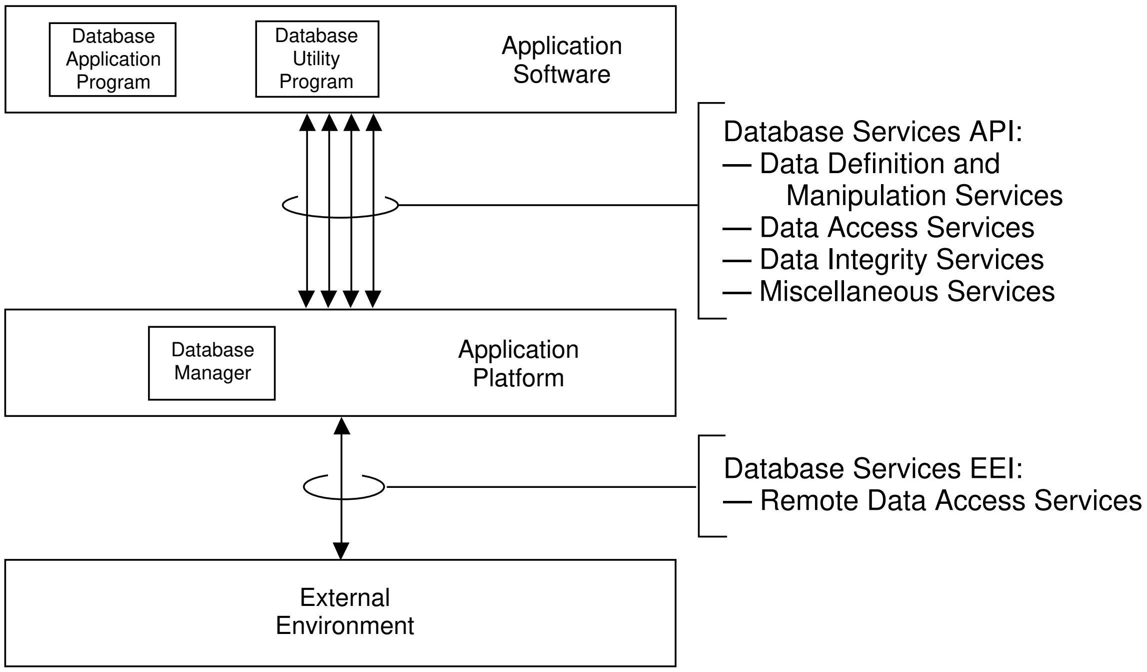

Figure 4-11 – POSIX Database Reference Model

sit on top of a POSIX OSE API, but there is very important value to the user in terms of portability if the database manager implementation does indeed sit on a POSIX API. This means that the database manager itself is portable. It should be noted that there will probably be implementations available of database managers that do not, in fact, sit on top of a POSIX API (or sit only partially on top of a POSIX API), that nonetheless provide the user the same database API. Such an implementation, using both POSIX API services and non-POSIX API services was described as ‘‘expansion’’ (see 3.6.1). Note that even though the model is drawn with only a single database manager, that does not imply that there may only be a single database manager available to an application. In fact, the coexistence of several database managers on the same system is consistent with this model, as is the ability of a single application to access two or more different database managers. The following extensions to the above model are specifically accommodated:

— There may be more than one database API. For example, there may be an ‘SQL’’API and an ‘ISAM’’API.

— There may be more than one database manager implementation for each different API. (For example, by two competing database vendors.)

— Applications may access more than one database manager.

This document has not described how a database manager is implemented in a POSIX Open System Environment, nor is it within the scope of this document to do so. It should be noted, though, that this model is very general and does not constrain the implementor. This model supports a number of varying real world implementation techniques, including:

— Application and database manager linked into a single process.

— Database manager consisting of more than one process.

— Database manager consisting of a client part linked into the application process and a server part running as a process on the same or another system.

4.4.4 Service Requirements

The Database Manager described in the previous subclause provides services to the Application Program via the Database API, and the Database Utility Programs provide other services (e.g., to human users such as a ‘‘Database Administrator’’). This subclause describes the service requirements of all service users of the system. It is intended to be a complete list of service requirements rather than examples. Database Services are the specialized data services required to create, access, and manage databases located on a processor node. Users of these services include end users and those charged with the ongoing management of the information processing and database infrastructure.

4.4.4.1 Application Program Interface Services

This subclause describes the major categories of database services available at the POSIX Application Program Interface (API). These services include:

— Data Definition and Manipulation Services

— Data Access Services

— Data Integrity Services

— Miscellaneous Services

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

Types of Data Objects
: Ability to perform the above operations on a variety of types of data objects, such as text, graphics, image, documents, and voice.

Types of Access Methods
: Ability to perform the above operations using a variety of access methods, such as indexed sequential access, nonindexed sequential access, and direct access.

Types of Database Management Systems
: Ability to perform the above operations on a variety of types of file and database management systems, and database management systems, such as relational, network, semantic, and object oriented

databases, and heterogeneous combinations of these database management systems.

4.4.4.1.1 Data Definition and Manipulation Services

These services relate to the ability of application programs to define and manipulate data. These services are:

— Data definition — ability to create, alter, or drop tables, views, records, fields, and/or data

— Data Manipulation — ability to insert, select, update, and delete tables, views, records, fields, and data

4.4.4.1.2 Data Access Services

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

— Data Query Facilities — ability to specify search conditions, consisting of a combination of select lists, predicates, and comparison operators

— Data Transparency — ability to transparently access data regardless of the location of that data.

— Remote Data Access — ability to access and update remote data

4.4.4.1.3 Data Integrity Services

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

— Locking — ability to specify locking of data to some degree of granularity

— Consistency — ability to specify and execute check and referential constraints that help ensure data correctness

— Transaction Control — ability to specify commit and rollback commands and guarantee serializability for database transactions E

— Synchronous Writes (Durability?) — ability to force the writing of data to nonvolatile storage

4.4.4.1.4 Miscellaneous Database Services E

Miscellaneous database services include: E

— Privilege Administration — ability to grant and revoke privileges for accessing and administering data

— Exception Handling — ability to have applications that are interrupted and notified of exception conditions, to receive control of the machine and take action in response to these exception conditions—even if the action is to ‘‘continue’’

— Screen Definitions — ability to create screen definitions, and define, display, and/or paint screens to communicate information about databases E

— Reporting — ability to create formatted reports.

— Dynamic Facilities — ability to temporarily turn control of a database to the end user for interactive access and manipulation of data, and then return control to the application.

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

4.4.4.2 External Environment Interface Services

External Environment Interface services are required for distributed database management systems. Also, to enable two or more databases to communicate with each other, a common interchange format is required. See 4.5.

4.4.4.3 Database Resource Management Services

These services are not visible to the application programmer at the Database API. These services are usually provided by Database Utility Programs. These services include:

— Database Administration Services

— Database Recovery Services

— Distributed Database Management Services

— Heterogeneous Environment Support Services

4.4.4.3.1 Database Administration Services

Database administration services refer to the ability for a designated data administrator to structure and configuration manage a database as a whole. The administrator allocates resources and monitors utilization to assure that authorized users receive the proper services. Archive functions, journaling, and logging services are available to the user and administrator on a selective basis. E

4.4.4.3.2 Database Recovery Services

Database recovery services refer to the ability to decide that there has been a failure, allow recovery from failure, and permit a slave copy to become a master copy. E

4.4.4.3.3 Distributed Database Management Services

Distributed database management services support the partitioning and partial replication of the databases.

4.4.4.3.4 Heterogeneous Environment Support Services

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

4.4.4.3.5 Flagger

A flagger is software that alerts programmers about any code that does not conform to the standard in question, such as the Structured Query Language standard.

4.4.5 Standards, Specifications, and Gaps

There are currently four database standards, either completed or under development. These are the relational data language SQL, a network data language called NDL, the Information Resource Dictionary System (IRDS) for data dictionary work, and a Remote Data Access (RDA) protocol. Table 4-5 summarizes the service requirements provided by the various standards.

Table 4-5 – Database Standards

Service	Type	Specification	Subclause
Data Definition and Manipulation Services Data Access Services Data Integrity Services	S	SQL: ISO 9075 ANSI X3.168	4.4.5.1
Data Definition and Manipulation Services Data Access Services Data Integrity Services	S	NDL: ISO 8907	4.4.5.1
Miscellaneous Services (Data Security and Integrity, Exception Handling, Screen Definitions, Reporting, Dynamic Facilities, Data Dictionary Services) Database Resource Management Services (Database Administration, Recovery From Failure)	E	IRDS: ISO DP 10027 N2642 (IRDS Framework), ISO DP 8800 N2132 (IRDS Interfaces), ANSI X3.138	4.4.5.2
External Environment Interface Services	E	RDA: ISO/IEC DP 9759	4.4.5.1

4.4.5.1 Current Standards

This subclause describes the current accepted standards that apply to this area.

SQL Standard Database Language

ISO 9075 (FIPS 127)

ANSI X3.168

ISO 9075 provides for many of the services described in 4.4.4, including Data Definition, Manipulation, and Integrity. It provides for two levels of compliance: the weaker Level 1 and the more capable Level 2. While ISO 9075 deals with SQL independently of programming language, X3.168 binds, or embeds, SQL within the programming languages COBOL, FORTRAN, Pascal, PL/1, C, and Ada.

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

NDL Standard Database Language

ISO 8907

ANSI X3.133

This standard, developed in 1981-1986 by the ANSI X3H2 Database Committee, specifies a data definition language (DDL) and data manipulation language (DML) for network model databases. This work is an outgrowth of the 1978 CODASYL specifications.

This standard provides for many of the services described in 4.4.4, including Data Definition, Manipulation, Access, and Integrity. The above services apply only to network databases (i.e., not to relational or semantic databases.)

No follow-on NDL activities are being conducted by ISO or ANSI.

4.4.5.2 Emerging Standards

This subclause describes the activities currently in progress to further standardize this area.

Remote Data Access (RDA) Protocol

ISO DP 9579-1	*Generic Remote Database Access* — DP 2
ISO DP 9579-2	*SQL Specialization* — DP 1

This standard, developed by the ECMA Technical Committee on Database Standards, TC22, submitted to ISO in 1985, specifies a protocol that allows remote access and updating, via OSI communications protocols, of relational databases or of database systems that support SQL.

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

Much work is planned on in this area by the ISO committee ISO TC97/SC21/WG3. A specific area of current interest is a generic RDA that uses a nonspecific database language (i.e., not SQL.)

Information Resource Dictionary System (IRDS)

ANSI X3.138	FIPS Pub 156, April 5, 1989
ANSI X3H4/90-28 (draft, 4 Apr 90)	IRDS Export/Import File Format
ISO DP 10027 N2642 (1988)	IRDS Framework
ISO DP 8800 N2132 (1988)	IRDS Services Interfaces

These standards are being developed by the ANSI X3H4 Database Group and the ISO/IEC /JTC 1/SC21 Working Group 3. Both groups are addressing the general area of data dictionaries, but, as described below, the emphases of the activities differ.

The ANSI group primarily addresses the user interface area; that is, how a human user can access the Data Dictionary Services described in 4.4.4.

The ISO groups concentrate more on the service interfaces (APIs) among lower level components and utilities of the database model.

Differences in scope and incompatibilities exist between the model being developed by ISO and the model approved by ANSI. They are independently developing the Services Interface, and an export/import facility.

4.4.5.3 Gaps in Available Standards

There are two significant areas described in 4.4.4 as requirements that are not addressed by standards:

— Methods to access data such as hashing and indexed sequential access to data is not currently standardized. There is no consensus in the standards community as to whether such standardization would be beneficial.

— Standardization of semantic and object oriented models have also not taken place, though groups like the ANSI Database system study group (DBSSG) are currently investigating the feasibility of standardization in these areas.

— I/O Services such as screen generation.

— Management and control of database services. Also include all gaps (all services without standards).

4.4.6 OSE Cross-Category Services

4.4.6.1 Security

The ability to specify logical database access control mechanisms is important to database security.

4.4.7 Related Standards

The standards and activities described in this subclause are related to the above and may also be relevant to your activities.

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

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

ISO/IEC 9804.3	(OSI CCR services)
ISO/IEC 9805.3	(OSI CCR protocol)
ISO 8824	*Information Processing Systems— OSI—Specification of Abstract Syntax Notation One (ASN. 1)*
ISO 8825	*Information Processing Systems— OSI—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN. 1)*

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

4.5 Data Interchange Services

Responsibility: Richard Scott

4.5.1 Overview and Rationale

The Data Interchange/Information Exchange components of the POSIX Open System Environment provide specialized support for the exchange of data between applications or components of applications. Without support for data interchange, problems can arise when attempts are made to move data between different operational environments or between two related applications. More specifically, data interchange problems arise in each of the five following situations:

— Movement of a single application program and its associated data between operational environments,

— Movement of data between cooperating application software within the same operational environment,

— Movement of data between cooperating application software operating in differing operational environments,

— Movement of data between related, but not cooperating, application software within a single operational environment, and across differing operational environments.

From the global view, the data interchange components can provide the means to satisfy the needs in each of these situations. These standards need to define physical formats, data formats, code sets, and data descriptions that are consistent across all implementations of the POSIX Open System Environment.

4.5.2 Scope

The data interchange component of the POSIX Open System Environment includes standard services, protocols, and data formats required to ensure that data can be interchanged between related application software. Physical media formats are beyond the scope of the POSIX Open System Environment.

4.5.3 Reference Model

The Data Interchange Services relate directly to the POSIX Open System Environment reference model that was presented in Figure 3-1. Figure 4-12 shows the components of the reference model that are significant for data interchange. The reference model defines the conceptual relationships required to provide these facilities. It should not be viewed as a description of an implementation. The key entities within the figure are the Application Software, the Application Platform, and the External Environment. To satisfy the data interchange service requirements, the POSIX Open System Environment must permit application software to transfer data to and from the external environment.

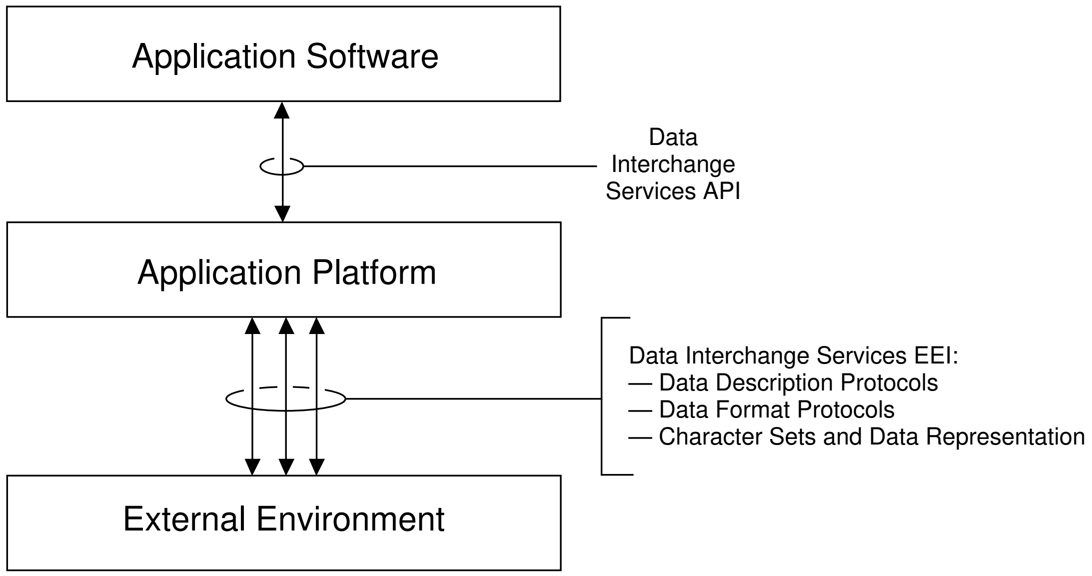

Figure 4-12 – Data Interchange Reference Model

The application software requests this transfer through the Application Program Interface. In response to those requests, the data interchange components of the Application Platform handle conversions to and from standard formats and the transfer of the information across the External Environment Interface (EEI). The EEI , which defines the format specifications required to support data interchange, can be divided into Data Description Protocols and Data Format Protocols. Data Description Protocols provide a means to identify the data that is present. Data Format Protocols provide the storage representation of the actual data.

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

4.5.4 Service Requirements

This subclause details the Data Interchange Services and protocols that are required to support application portability and interoperability. Subclause 4.5.4.1 describes the API service requirements. 4.5.4.2 describes the EEI service (i.e., protocol) requirements.

Data interchange is one of the components of the POSIX Open System Environment that is now just beginning to evolve. At this time, the general requirements for services are understood, but there is little general existing practice that can be pointed to as showing that current service requirements are both necessary and complete. Most existing practice is limited to a specific application domain. As a developing area, data interchange represents gaps in both the definition of service requirements and standards. The data interchange component is, none the less, critical for supporting application portability and interoperability. The data interchange service requirements are currently described to the extent possible at this time in their evolution. More detail will be added in future revisions of this guide.

4.5.4.1 Application Program Interface Services

The API services to support data interchange need to provide the ability to store and retrieve data using the formats and protocols provided at the data interchange EEI.

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

4.5.4.2 External Environment Interface Services

This section identifies the EEI services required to support data interchange. These services are all in the form of protocol and format definitions. As shown in Figure 4-12, these protocols include:

- Character Sets and Data Representation
- Data Format Protocols
- Data Description Protocols

These protocols are required to support the exchange of information between application software entities, both within a single application platform and between application platforms.

4.5.4.2.1 Character Sets and Data Representation

The ability to support Character Sets and Data Representation is crucial to providing effective data interchange between application software operating under differing language and cultural conventions. These services add facilities to the POSIX Open System Environment to identify the character set and data representations associated with textual data. A detailed description of the requirements in this area can be found in 5.1.

4.5.4.2.2 Data Format Protocols

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

— Detailed storage format for the value

— The data value in an environment-neutral format

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

4.5.4.2.3 Data Description Protocols

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

4.5.5 Standards, Specifications, and Gaps

See Table 4-6.

4.5.5.1 Current Standards

Open Document Architecture (ODA)/Open Document Interchange Format (ODIF)

The ODA/ODIF standard (ISO 8613 Parts 1-8) provides a standard for the structures used to represent documents. The ODA model defines a comprehensive description of a documents format. It supports both reproduction of the original document and also editing and formatting of the document.

Part 5 of the ISO ODA standard specifies the interchange format for ODA documents; specifically ODIF. ODIF is an ASN.1 (ISO 8825) based presentation of the ODA document.

Standard Generalized Markup Language (SGML)

SGML (ISO 8879) is a language that allows users to precisely define the structure of a document. The key difference between SGML and ODA/ODIF is that the former provides the flexibility to define custom document types.

Table 4-6 – Data Interchange Standards

Service		Type	Specification	Subclause	
					E
Data Description Protocols	ODA/ODIF	S	ISO 8613 Parts 1-8	4.5.5.1	E
	SGML	S	ISO 8879	4.5.5.1	E
	EDIFACT	S	ISO 9735	4.5.5.1	E
	STEP	E	ISO DP 10303	4.5.5.2	E
	EDIFACT	S	ANSI X.12	4.5.5.1	E
	IGES	G	NBSIR 86	4.5.5.3	E
	VHDL VHSIC	G	IEEE P1076	4.5.5.3	E
Data Format Protocols	ODA/ODIF	S	ISO 8613 Parts 1-8	4.5.5.1	E
	SGML	S	ISO 8879	4.5.5.1	E
	CGM	S	ISO 8632	4.5.5.1	E
	CGM	S	ANSI X3.122-1986	4.5.5.1	E
	STEP	E	ISO DP 10303	4.5.5.2	E
	EDIFACT	S	ISO 9735	4.5.5.1	E
	EDIFACT	S	ANSI X.12	4.5.5.1	E
	IGES	G	NBSIR 86-3359	4.5.5.3	E
	VHDL VHSIC	G	IEEE P1076	4.5.5.3	E
	CDIF	G		4.5.5.3	E

Computer Graphics Metafile (CGM)

CGM (ISO 8632, ANSI X3.122-1986) provides a standard means for the storage and exchange of computer graphics. Graphic information is stored in a device- and resolution-independent fashion that can support both the display and the manipulation of the data.

Electronic Data Interchange (EDI)

The EDI standards [ISO 9735 (EDIFACT), ANSI X.12] provide support for the exchange of structured business data. EDI is typically used to transfer business documents such as purchase orders, invoices, promotional announcements, and electronic funds transfer information.

E

4.5.5.2 Emerging Standards

Standard for the Exchange of Product Model Data (STEP)

STEP (ISO DP 10303) is a neutral mechanism capable of completely representing product data throughout the life cycle of a product. The completeness of this representation makes it suitable not only for file exchange, but also as a basis for implementing and sharing databases of archiving.

4.5.5.3 Gaps in Available Standards

4.5.5.3.1 Public Specifications

Most standards activity in the data interchange area has been isolated to specialized application areas. These activities have attempted to support data interchange by limiting the scope of the effort to a specific type of data. These industry standards include:

Initial Graphics Exchange Specification (NBSIR 86-3359)

IGES is used heavily in the exchange of graphical information between applications.

CASE Data Interchange Format (CDIF)

The CDIF Technical Committee is developing a data interchange format to serve as an industry standard for exchanging information between Computer-Aided Software Engineering (CASE) tools. CDIF is an EIA-endorsed initiative. It assumes that two or more tools may interface asynchronously with each other and will transfer information from one to another via ''CDIF files.'' The types of information that may be contained in these files is defined by the CDIF Conceptual Models.

Hardware Description Language (VHDL VHSIC)

The VHDL standard (IEEE P1076) defines a representation for the exchange of CAD representations of electronic circuits.

4.5.5.3.2 Unsatisfied Service Requirements

None of these standards addresses a general means to handle application data in a manner to ensure portability between environments. The closest attempt is the effort just beginning in POSIX.8 to define a means within the network interface to provide data portability. However, even this effort is not attempting to solve the broader issue of interoperability of multiple applications. The existing standards have all evolved to support the interchange of specific types of data between separate applications. Support for general data portability is not addressed by existing standard, except for ISIS, which does not appear to have caught on.

4.5.6 OSE Cross-Category Services

Not applicable.

4.5.7 Related Standards

The following standards are related to the services covered in this clause as they address some of the services described in section 4.6.4 at some level. Each of these related standards are addressed fully as part of another service category.

ASN. 1	ISO 8824	Abstract Syntax Notation (Clause 4.3)
	ISO 8825	ASN. 1 Basic Encoding Rules (Clause 4.3)
MHS	ISO/CCITT X.400-1984	Message Handling System (Clause 4.3)
	ISO/CCITT X.400-1988	Message Handling System (Clause 4.3)

4.5.8 Open Issues

Data interchange support must address hardware/software differences between environments. The key concerns in transporting data that must be addressed will include the character set, word size, and byte ordering for the operating environment along with a accurate identification of the data value.

The data portability standards adopted within POSIX Open System Environment need to define data formats that will enable applications to create data that can be used read and properly interpreted on differing operating environments and by other application software.

4.6 Transaction Processing Services

Responsibility: *Bob Gambrel*

4.6.1 Overview and Rationale

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

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

The following is quoted with a few editorial changes from ISO/IEC DP 10026-1, the ISO Distributed Transaction Processing standard draft:

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

4.6.2 Scope

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

— Transactional access to a single database manager on a single machine

— Transaction access to nondatabase ‘‘resource managers’’ (such as the software managing the cash in an automatic teller machine)

— Distributed Databases—databases spanning multiple machines, but accessed by application programs as if just a single database

— Online Transaction Processing (OLTP)—the scheduling of ‘‘transaction programs’’ based on terminal input with consolidated recovery of the database updates and the terminal messages

— Distributed Transaction Processing (DTP)—different machines running multiple application programs with multiple databases, using a client/server or conversational application-to-application communications paradigm

Note that Transaction Processing Services are used in all of the above situations, and others too.

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

This clause addresses the current work in progress in groups such as ISO and others.

4.6.3 Reference Model

This subclause addresses the conventional Transaction Processing Reference Model, the POSIX OSE Reference Model (incorporating transaction processing considerations), and other important real world considerations introduced by Transaction Processing.

4.6.3.1 Conventional Transaction Processing Reference Model

A model for transaction processing is developed here to complement the POSIX system model. Current work in progress by the POSIX.11 Transaction Processing Working Group and other groups such as ISO/IEC JTC 1/SC21 Open Systems Interconnection—Distributed Transaction Processing may result in a Transaction Processing Reference Model more suitable than the one developed here. At that time, such a model will be referenced and incorporated into the POSIX OSE reference model. Until that time, the current model will be used as a convenient means for describing the services needed in this domain.

While transaction processing services have usually been thought of as applying to databases, the applicability goes further. Nonetheless, the description given here of the transaction processing model shows how the transaction processing program can view the transaction services as an extension of the the Database view of the POSIX OSE reference model as shown in Figure 4-10. From the transaction application program point of view, a transaction processing system has additional capabilities that go beyond those provided by database systems. These services to the transaction application program are provided at an API that is called the *Transaction Manager API.* (See Figure 4-13.) For convenience in discussing the model, the provider of those services is called the *Transaction Manager* (TM).

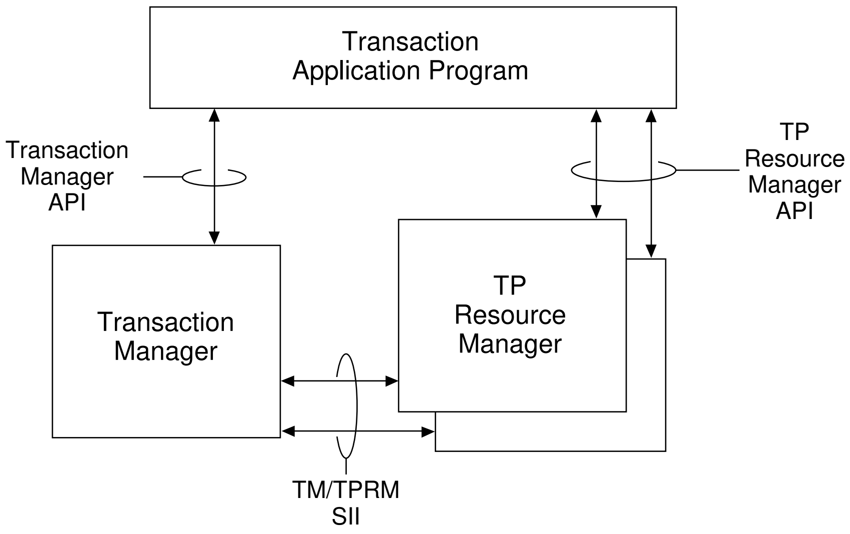

Figure 4-13 – The Conventional Transaction Processing Model

The transaction application program requests services provided by the *TP resource manager* [2)] (e.g., a database manager) via the *TP resource manager API.* The transaction manager API and the TP resource manager API are called the *transaction services API* and provide all the services needed by transaction application programs.

The ACID properties are maintained for each managed resource by a *TP Resource Manager (TPRM),* coordinated by a *Transaction Manager.* The interface between the TP Resource Manager and the Transaction Manager will be called the *Transaction Manager/ TP Resource Manager SII* (*TM / TPRM SII*).

The ACID properties can be applied not only to resources such as databases, but also to other resources that might not be obvious. For instance, a transaction that dispenses cash may wait until the cash dispensing machine has signaled completion before considering the transaction complete and updating involved accounts.

---

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

This illustration also shows the limits of transaction processing resource management. The machine could signal completion, but a mechanical problem could prevent the cash from being dispensed correctly, undetected by the system.

Besides database TPRMs and miscellaneous nondatabase TPRMs, a third class of of TPRMs exist: Communications TPRMs (cTPRM). Services provided by cTPRMs are used when two cooperating transaction application programs need to communicate with each other in the context of the same transaction. At least two communications paradigms have been identified as beneficial to cooperating transaction applications programs: client/server (RPC, single request/response) and conversational (peer-to-peer, dialog).

4.6.3.2 POSIX OSE Reference Model (with Transaction Processing)

The conventional transaction processing model is shown integrated into the POSIX OSE Reference Model in Figure 4-14. Because the POSIX OSE Reference Model does not address System Integration Interfaces (SIIs) per se, they are not shown in the integrated model. What is shown are the transaction processing services APIs and EEIs.

4.6.3.3 Implementation Aspects

The POSIX OSE Reference Model does not provide for a way to expose the details of the Application Platform. System Internal Interfaces (SIIs) are beyond the direct scope of this guide because they do not directly affect application portability or system interoperability. In the Transaction Processing world, as shown in the conventional Transaction Processing Reference Model (see 4.6.3.1), the existence of Transaction Managers and multiple TP Resource Managers connected by the TM/TPRM SII is important. One way to think about the real world implications of this is that TP Resource Managers and the Transaction Managers could both be implemented in the Application Platform as separate entities, connected to each other by the TM/TPRM SII. This does not, however, imply that the two must be implemented as separate entities, though there are advantages to the user if they are separate.

NOTE: For application portability it is not required that the application platform actually consist of Transaction Managers and TP Resource Managers, but in the new age of Open Systems, there are clear advantages in doing so. Two advantages seem obvious: the ability to ‘‘mix and match’’ Transaction Managers and TP Resource Managers from different vendors; and the ability of a user to construct his/her own TP Resource Manager to manage particular critical resources. A market has already developed for ‘‘plug compatible’’ TMs and TPRMs. All TPRMs at this printing are Database type TPRMs. It is expected that a market will also develop for Communications type TPRMs. It is not at all clear that the industry will develop other types of widely applicable TPRMs, thus forcing users to develop their own. Users could use the interface described here to do such development work.

This NOTE very briefly describes the services that should be provided at such an interface.

The TM/TPRM interface must provide the ability of TMs and TPRMs to: register with each other; obtain recovery status information; pass along transaction identifier information; rollback, prepare to commit, and commit the transaction. The interface must provide for the needs of the full range of transaction processing including distributed transaction processing with multiple TPRMs.

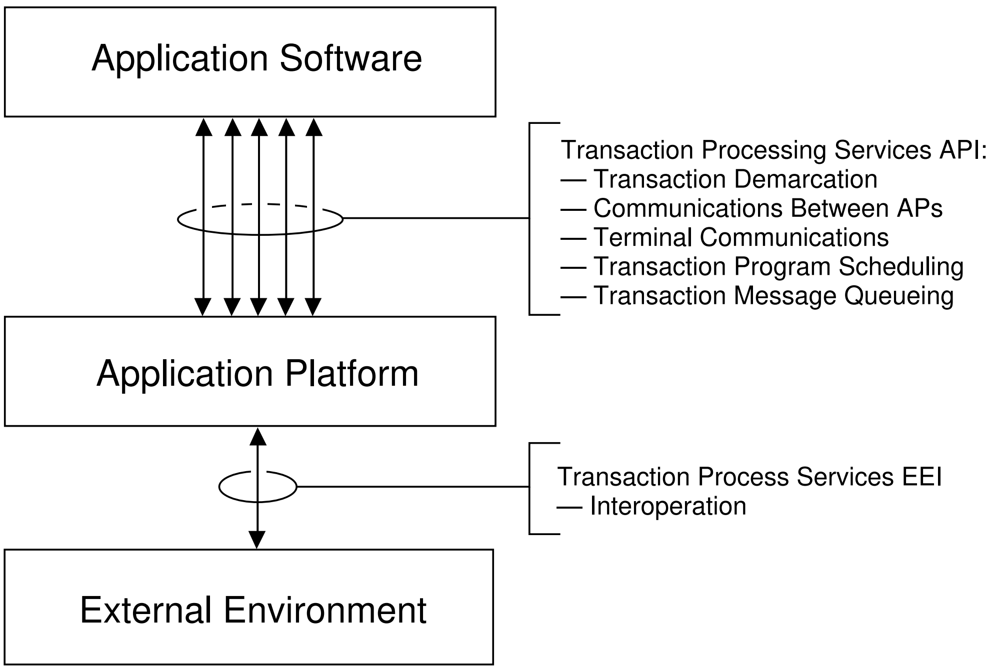

Figure 4-14 – POSIX OSE Transaction Processing Reference Model

Finally it should be noted that the industry recognizes the need for standardization of components as well as the standardization of portability and interoperability. At least one industry group is standardizing and several vendors are implementing products utilizing an interface as described here.

4.6.4 Service Requirements

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

4.6.4.1 Application Program Interface Services

E

The Transaction Services API provides various services to the application programmer:

Transaction Demarcation

— Indicate the start of a transaction.

- Indicate a transaction has ended successfully (commit) or unsuccessfully (rollback).
- Indicate the beginning and ending of nested ''subtransactions'' whose commitment is independent of the ''parent transaction''. (Nested within a parent transaction can be multiple subtransactions. Subtransactions are independent of each other, and whether subtransactions commit or not does not affect the commitment of the parent.)
- Suspend and resume transaction mode (to do work which is not be committed or rolled back when the transaction is completed). This can be thought of as nesting nontransaction work within a transaction.

E

Communications Between Transaction Application Programs

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

NOTE: The above services provide ''Distributed Transaction Processing.'' E

Terminal Communications

- Send and receive messages to and from terminals within the context of a transaction (i.e., messages sent to terminals are not to be actually delivered unless the transaction commits).

Transaction Program Scheduling

- Cause to be started another transaction application program outside of the context of this transaction. Involved here are two transactions: one starts the other. The actual scheduling of the second transaction can be dependent on the completion or not of the original transaction.

Transaction Message Queuing

- Define a ''message'' (based, possibly, on screen input from the end user) that, from the application point of view, precisely defines a unit of work to be done by this transaction or another transaction.
- ''Send'' a message to another transaction application program.
- Retrieve the next message (and then act upon it)
- Prioritize and associate start times with messages

  NOTE: The actual handling of messages can be dependent on the completion or not of the original transaction.

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

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

NOTE: TP Resource Managers, in general, ‘‘protect’’ a critical resource. Databases are good examples of TP resource managers where the resource actually being protected is the data, for example, of an enterprise. Often the data of an enterprise reflects the amount of a real resource such as cash holdings. In this case a tangible resource is indirectly protected by a TP resource manager. The importance to the enterprise in insuring that the data (quantifying money) is accurate should be obvious. Other TP resource managers, on the other hand, could protect an actual, tangible resource. An example of such a TP resource manager is the program that controls the cash drawer of an automated teller machine. The resource protected is the cash in the drawer. The actual application program interface of the TP resource manager protecting that resource could include the ability to reduce the amount of money in the drawer (by pushing it out of the machine). A transaction application program using two TP resource managers (a conventional database manager that keeps track of the balance in accounts, and the teller machine’s cash drawer TP resource manager) would want to insure that the two TP resource managers decrement both the cash and the balance of the correct account in the context of a single transaction (i.e., with the ACID properties.)

The TP Resource Manager API, then, generally provides the following services:

— Increment or decrement a valuable resource by a certain amount.

— Determine the amount of a valuable resource that remains.

Specific capabilities for the very wide variety of specific TP resource managers, cannot, of course, be documented here.

4.6.4.2 External Environment Interface Services

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

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

4.6.4.3 OLTP Resource Management Services

The services listed in this subclause are not provided by application program interfaces or external environment interfaces.

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

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

- — Modeling Services — Predict the system resources needed to process a given transaction processing workload.
- — Directory/Namespace Services — Map names to addresses.
- — Recovery/Restart Services — Recover and restart transactions involving one or more transaction application programs using one or more TP Resource Managers.
- — Test Services — Automatically generate tests for workload simulation, etc.
- — System Configuration Services — Replace or add transaction application programs without the need to shut down the execution environment.

4.6.5 Standards, Specifications, and Gaps

There are currently three transaction processing standards development activities, either completed or in the draft stage. Table 4-7 summarizes the service requirements provided by the various standards.

Table 4-7 – Transaction Processing Standards

Service	Type	Specification	Subclause
API Services	E	IEEE P1003.11	4.6.5.2
EEI Services	E	ISO/IEC 10026-1, -2, -3	4.6.5.2
Resource Management Services	G	–	4.6.5.3

Table 4-8 summarizes the applicability of the various standards to the various programming languages supported by the POSIX Open System Environment.

4.6.5.1 Current Standards

None.

4.6.5.2 Emerging Standards

OSI Distributed Transaction Processing (DTP)

ISO/IEC DIS 10026-1
ISO/IEC DIS 10026-2
ISO/IEC DIS 10026-3

These standards, developed by ISO/IEC JTC 1/SC21/WG5, deal expressly with the OSI services and protocols for transaction mode communications in an OSI environment.

Table 4-8 – Transaction Processing Standards Language Bindings

Standard	LIS	Ada	APL	BASIC	C	C++	
POSIX. 11	E				E		E
Standard	**COBOL**	**C-LISP**	**Fortran**	**Pascal**	**PL/1**	**Prolog**	E
POSIX. 11							E

NOTES: LIS — Language-independent specification is available. E
Ada, APL, BASIC, — Language-dependent specifications exist. E
S, E, G — Standard, Emerging Standard, Gap E

These standards provide for some of the communications services described in 4.6.4.1.

POSIX. 11 POSIX Transaction Processing

POSIX. 11

The POSIX. 11 working group, formed in 1989, is chartered to work on a profile for Transaction Processing within the POSIX OSE. In the process of developing that profile, it has identified a number of gaps in the standards coverage and is in the process of proposing base standardization activities to address those gaps. Specifically, P1003.11 is currently working on the following services identified earlier:

- Transaction Manager (TM) Services provided at the Transaction API.
- Services provided at the Transaction Manager/ TP Resource Manager (TM/TPRM) SII. A typical TPRM is a database manager (e.g., SQL).

POSIX. 11 is working to assure that POSIX Transaction Processing work is consistent with the emerging work of OSI DTP (cited above), certain ongoing work of X/Open TP, several related POSIX activities (POSIX .1 {2}, POSIX. 4, POSIX. 8) and the work of ANSI X3T5.5 (RPC).

4.6.5.3 Gaps in Available Standards

4.6.5.3.1 Public Specifications

Existing standards and emerging standards do not adequately address all the requirements identified earlier. While POSIX. 11 is addressing some of the gaps, there are many other gaps still not being addressed by formal standards committees. Most notable is the work of X/Open TP. While not formally a standards making body, it is addressing most of the gaps, and its output will be potentially useful as the basis of a formal standard.

X/Open TP

This group published an ''Online Transaction Processing Reference Model'' in 1987 and in 1990 published ''Preliminary Specification—Distributed Transaction Processing: The XA Specification.'' The group is studying the use of OSI DTP, two-phase commit, and global transaction identifiers. The group is also actively exploring APIs in support of peer-to-peer distributed transactions.

The work of this group addresses several of the services addressed in this clause, including transaction demarcation and conversation services.

Consideration is also being given to allowing alternative interoperability standards including proprietary mechanisms. Additional APIs are being defined by X/Open TP to facilitate this.

4.6.5.3.2 Unsatisfied Service Requirements

Other than the work of X/Open TP, the following areas are not currently being addressed by standardization activities: communications, terminal communications, program scheduling, message queueing, management, monitoring, modeling, directory/namespace, recovery/restart, test, and system configuration.

4.6.6 POSIX OSE Cross-Category Services

Not applicable.

4.6.7 Related Standards

CCR

The following standards relating to commitment are related to the ISO/IEC DIS 10026 series and are referenced in DIS 10026:

ISO/IEC DIS 9804-3
ISO/IEC DIS 9805-3

See 4.3 for more information.

E

SQL Standard Database Language

The following standards for SQL also provide transaction demarcation services for relational database access:

ANSI X3.135.1 (ISO 9075, FIPS 127)
ANSI X3.168

See 4.4

4.7 Graphical Window System Services

Responsibility: Marti Szczur and Ruth Klein

Editor's Note: Variations on the term ''human computer interaction'' and HCI in this clause have been replaced globally by ''graphical window systems'' without further diff marks. ''Human user'' has also been replaced by ''user.''

4.7.1 Overview and Rationale

The graphical window system interface is a key component of computer systems that support direct user-machine interaction. Until recently, most computer operating systems interpreted commands that were typed in from the keyboard of an alphanumeric computer terminal. Special purpose applications, such as those for CAD/CAM, have always presented user interfaces based on series of menus or pointing at visual displays with tablets and lightpens. The availability of low-cost bitmapped graphic workstations and personal computers has led to the proliferation of graphical user interfaces (GUI s), windowing technologies, generic commands, and an assortment of selection techniques (e.g., mouse, track ball, tablets). In several of these technologies de facto standards are emerging and becoming informally accepted by the user community, and with more frequency, mandated for use in systems being developed within government agencies and private industry. The primary motivations for considering graphical window system standards and their relation to POSIX standards include:

- The existence and popularity of windowing systems
- The requirement for development of applications that take advantage of the windowing system environment
- The requirement of many users and manufacturers for a basic consistency in the presentation and behavior of graphical window systems across multiple graphics platforms

As the windowing system technology evolves within the graphics environment, the differences between windowing services and graphic services becomes less distinct. The distinction for purposes of this document is that graphic services are associated with providing general purpose interfaces for creating virtually any kind of two- and three-dimensional graphics (e.g., GKS for 2-D and PHIGS for 3-D). Graphical window system services certainly utilize graphic technologies, but are limited to providing graphics related to window-based user interfaces and specifications on how users may interact with an application within a window environment. The graphic services are addressed independently in 4.8.

4.7.2 Scope

Standards and standards initiatives in the graphical window system interface area cover a wide area, ranging from keyboard layout to screen management. In this clause, the following specific standards are considered:

— Protocols for window management on a local or remote display device

— Application Program Interfaces (API) for such protocols

— Graphical window system drivability features that define a common subset of ‘‘look and feel’’; i.e., appearance, screen positioning, and behavior of graphical window system objects within windows on a graphic screen

— Language-independent functional specifications and appropriate associated language bindings for the display, manipulation, and management of interaction objects within windows on a graphic screen

— Command-language interfaces that may be entered interactively by the user or retrieved from a stored procedure.

— Language-independent functional specifications and appropriate associated language bindings required to support character (non-bitmapped) terminals.

— Language-independent functional specifications and appropriate associated language bindings for the translation, manipulation, and management of command statements (or messages).

Standards relating to the following are not considered:

— Graphics; see 4.8.

— Keyboard layout (out of scope for graphical window system services)

— Network transport protocols; see 4.3.

— Hardware device interfaces (out of scope for graphical window system services)

4.7.3 Reference Model

This subclause identifies the entities and interfaces specific to the construction of a graphical window system architecture. This architecture is consistent with, and extends the architecture of, Section 3. As illustrated in Figure 4-15, the interface components involved in the user interface process are divided into two groups, called the external environment interface (EEI) and the application program interface (API).

The EEI is concerned with the communication with the user via the physical graphical window system devices (e.g., keyboard terminal, mouse, display screen). The applicable EEI standards are driven primarily in support of user and data portability across different application platforms. Standards and guidelines are intended to define a minimal set of commonality in graphical window systems,

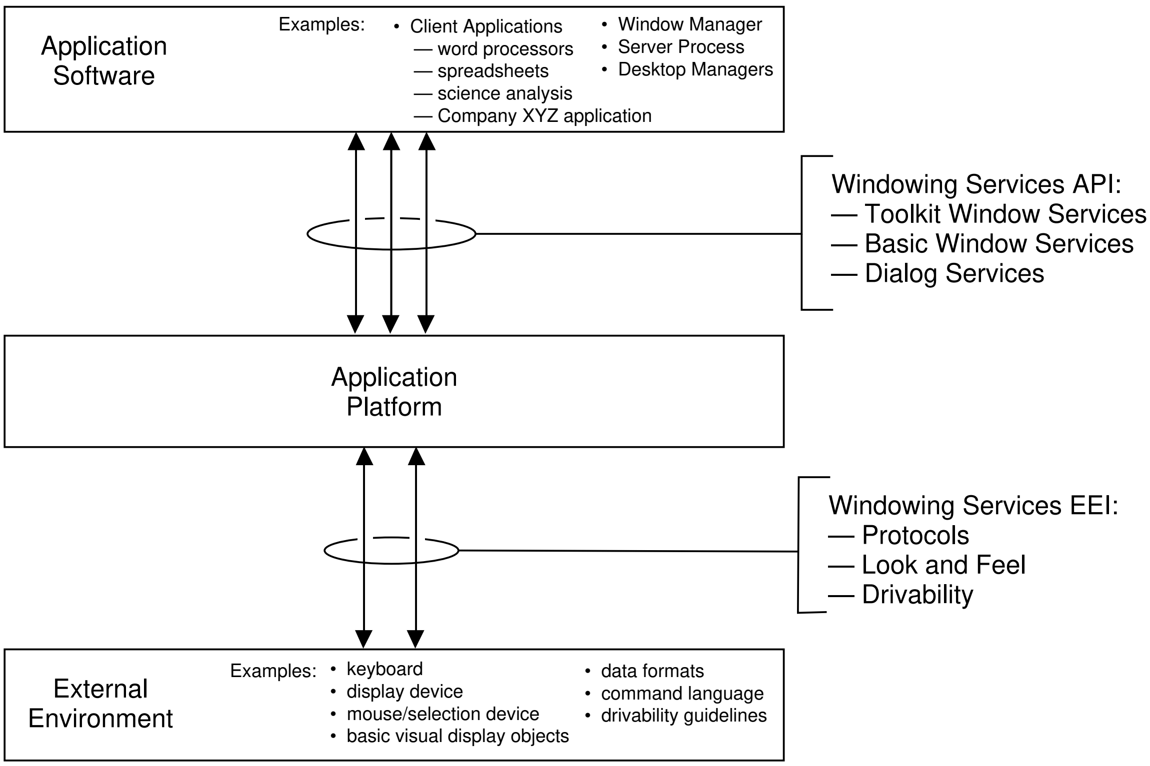

Figure 4-15 – Windowing Reference Model

which will eliminate problem areas such as:

— Error provoking inconsistencies

— Misleading expectations about the results of user actions

— Gross inconsistencies in the high level user model or metaphor

— Incompatible motor control tendencies

The drivability concept derives its name from the concept of ''driving'' an interface. A frequently cited analogy is the automobile. Having a standard location for the clutch, brake, accelerator pedals, ignition key, and steering wheel allows a driver to move between car models with relative ease (until he/she has to roll down the window, turn on the lights or windshield wipers!) Similarly, the EEI drivability guidelines will provide standards for graphical window systems that will ensure ease of moving between application platform models. For example, which mouse click causes an interaction object (e.g., radio button) to be selected or how a scroll bar should behave would be candidates for standard EEI specification.

The API is concerned with the interface between the application semantics and the graphical window system services. It is the interface between the application software and the application platform and is defined primarily in support of application portability. These services provide functions for creation and manipulation of visual display objects such as menus, buttons, scrollbars, and dialog boxes. In addition, these functions allow information about user actions to flow back to the application software; for example, when the user has selected an item from a menu. This information about user actions is known as an event. Applications that require communication with the user are inherently event-driven. That is, associated with an application's dialog window (i.e., a window in which a user response is expected) is a main event loop waiting for the user to make a selection that will trigger an operation to be performed by the application.

The API will support a specific user interface policy, which will define the application's ''look and feel.'' Although the specific look and feel need not be standard across application platforms (i.e., different implementations of the API may have unique styles) the API definition shall ensure that the application software can be ported across POSIX platforms; and the API shall support the EEI drivability guidelines, enabling users to easily operate the application across platforms.

Elements of the graphical window system architecture are Application Software Elements, Application Program Interface (API) elements, and External Environment Interface (EEI) elements. These elements are linked by the use of common concepts and definitions associated with the graphical window system entities, interfaces, services, and standards.

4.7.3.1 Application Software Elements

Application Entity Elements include:

(1) Window System Server

The Window System Server provides a function that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Applications and other programs that use basic windowing services are called ''clients.'' Many clients may talk to the same server. All application requests to write to the screen must go through the server via the basic windowing services. The server is independent of operating system, programming languages, or network communication.

(2) Window Manager

A Window Manager provides a uniform method for manipulating windows, which includes a basic set of window management capabilities that allow for development of alternative and/or user-preferred window managers. Required graphical window system capabilities shall include, but are not limited to:

— Resize window

— Move window

— Push/pop window to top/bottom

  - — Shrink window to a reduced visual representation of window (i.e., frequently referred to as an icon of the window)

(3) Local and Remote Applications

These applications are clients that provide the functions required to perform the specific task(s) that the user needs to achieve (e.g., spreadsheets, scientific analysis systems, CASE tools, process and control tasks.)

4.7.3.2 Application Program Interface (API) Elements

The API are language binding specifications that define the services available to the application programmer. API Elements are: basic window services, toolkit window services, and dialog services.

4.7.3.3 External Environment Interface (EEI) Elements

The EEI elements are specifications (and in some cases, aspects of physical objects) that define how the application platform interacts with the external world. Note that application software, as defined here, interacts with the outside world only via the application platform.

External Environment Interface Elements include:

- — Display Device Specifications
- — Data Protocol Format
- — User Drivability Guidelines (e.g., ‘‘look and feel’’ of window interface)
- — Keyboard Device Specification
- — Selection Device Specification (e.g., mouse, graphics tablet, touch screen)
- — Command-language Definition (syntax and semantics guidelines)

4.7.4 Service Requirements

Graphical window system services provide a controlled interface between the application-specific software and the user-interface-specific software, allowing each to be designed and implemented separately. Users of these services include all POSIX system users and those charged with maintaining the processor and graphical window system communication. A common, standardized graphical window system for applications should be available to users across all POSIX Open System Environments.

Services shall support raster (i.e., bitmapped) graphics displays. Methods for supporting vector graphics displays can be addressed, but are not mandatory.

4.7.4.1 Application Program Interface Services

Application services include those services made available to the application developer to separate the application functions from the graphical window system functions as much as possible. They include such areas as screen management, windowing, and user input device services.

These standard services support requirements for application portability, software commonality, application interoperability and data communications transparency.

A programmer may access the following services for an application via language bindings. E

4.7.4.1.1 Basic Window Services

The basic window services, callable from client applications, support a window-based user interface. They should be based on a ''client-server'' model. The server is a program that handles communication connections from clients, demultiplexes graphics requests onto the screens, and multiplexes input back to the appropriate client. Many clients may talk to the same server. All application requests to write to the screen must go through the server via the basic windowing services.

The major functional areas are:

— Window Management

— Presentation Management

— Event Handling

— Error Handling

— Interclient Communications

— Input Device Management: Keyboard, Pointing Device

— Screen Management

— User Preferences Management

— Server Connection Management

The following functions are available under each functional area.

Window Management

Functions available for Window Management are:

— Create a window, map a window onto the screen, delete a window (includes support for character-based emulator window)

— Manipulate a window (move, resize, change view precedence)

— Manipulate window attributes (set, get, change; attributes may be related to appearance, redraw performance, event handling, or change authority)

— Seize and relinquish control over the Server for display purposes (permits uninterrupted client output; output requests from other clients will be queued and displayed later)

Presentation Management

Functions available for Presentation Management are:

— Associate data with a window (context manager functions and association table functions)

— Manipulate the graphics context for a given object (create a graphics context, obtain current graphics context, change graphics context) E

— Get and set fonts (load font, list fonts, unload font) E

— Draw graphics primitives (draw arc, draw line, fill rectangle, clear rectangular window, clear entire window) E

— Manipulate window cursors (create, destroy, assign, change) E

— Draw text and obtain text metric information

Event Handling

The basic window services support application requirements to respond to the user's actions, rather than forcing the user to respond to the application in a rigid, serialized manner. This requirement necessitates that a program either (1) be capable of handling any one of a number of events at any single point in time, or (2) attach a routine to each event to be called automatically when that event occurs. There is a separate set of events for each window used by the application. An application selects the events for a particular window, maps the selected events to the window, and reads events from the event queue as they occur. There are three major types of events:

— Input device events (button press event, keypress event) E

— Window management events (window exposure event, colormap event) E

— Client message events (selection data transferred (by another application) event, private interclient communication event) E

Functions available for Event Handling are:

— Select events

— Map events to a window

— Get information about events

— Send events

Error Handling

Functions available for Error Handling are:

— Get error message

— Get error description

— Set error event handler routine

Interclient Communication

The basic window services are required to be network transparent to an application or client. This means that an application on one host may write to the display screen connected to another host without being aware that networking is involved. The basic window services handle the network connections and follow the protocols necessary for the application to interact with the display. This convention allows redistribution of applications in a networked system with no effect on the application software. Therefore, an application client cannot assume that another client can open the same files or seize the same processing environment. Interclient communication via the server has three forms:

— Properties

Clients may associate arbitrary information with a window; generally used for communication between a client and the window manager.

— Selections

Selections are selected by the user out of one client's window, then ''sent'' to another client and displayed in the second client's window.

— Cut Buffers

Cut Buffers are a specialized form of communication. It is possible to receive notification when a cut buffer (property) is set.

Functions available for Interclient Communication are:

— Manipulate window properties (list, delete, change, get) E

— Set and get selections

— Manipulate cut buffers

Input Device Management

Functions available for Input Device Management are:

— Receive keyboard input and pointing device button events

— Gain exclusive control of keyboard or pointing cursor

— Track the pointing cursor

— Change Server-wide keyboard mappings

— Set and get keyboard and pointing device preferences

Screen Management

Functions available for Screen Management are:

— Manipulate color using colormaps (copy, change, install, deinstall, get default) E

— Get, display, and manipulate bitmapped screen images

— Screen saver functions (blanking screen on idle)

— Retrieve display information (default colormap, number of display planes, screen width and height) E

User Preferences Management

The services and data structures used for managing user preferences are provided and collectively referred to as User Preferences Management. There may be up to four sets of options that need to be read and merged: E

— The user's defaults stored in the root window's user resource manager property

— The user's defaults stored in a user's defaults file

— The application program's defaults

— The command line arguments

Functions available for User Preferences Management are:

— Set and get preference data

Server Connection Management

Functions available for Server Connection Management are:

— Control access to the Server [add host to the access control list (ACL), list ACL, disable ACL] E

— Connect and disconnect a client from a Server (and the display controlled by the Server)

— Obtain Server implementation information

— Flush output buffer to Server and wait for Server to process all events in the output buffer

4.7.4.1.2 Toolkit Window Services

The Toolkit Window services provide a mechanism for runtime access to a library of visual objects. A visual object is a graphical display object (i.e., interaction object) with associated software that receives input from users (typically via a keyboard and a pointing device) and communicates with applications and other visual object software. The graphical representation of a visual object can be modified to reflect the results of application processing. Examples of visual objects are graphical push buttons, check boxes, and editing boxes. (Note: The term used within the X Window System community to define visual objects is ‘‘widgets.’’)

Toolkit Window services are provided for two reasons:

- To allow application software to directly utilize a visual object library
- To allow application-specific visual objects to be created and added to the widget library (Note: creating a visual object includes writing software that uses the Toolkit services)

Therefore, Toolkit services may be logically divided into two categories, with some overlap: Visual Object Interface Services, which are called by an application or dialog service, and Visual Object Programming Services, which are called by the visual object software.

An application may use Toolkit Window services to:

- Perform toolkit initialization/exit
- Set up visual object resources
- Create/delete a visual object
- Display a visual object
- Add/remove application-specific routines to be called by a visual object (event callbacks)
- Retrieve/modify the state of a visual object
- Turn control over to the toolkit for user input processing

A visual object software program may use Toolkit Window services to:

- Manage child visual objects (a child visual object is a visual object that is displayed inside of another visual object)
- Manage window events, timer events, and file input events
- Handle visual object geometry (sizing, positioning, child visual object placement)
- Handle user input
- Manage visual object resources
- Translate an event into an action
- Manipulate graphics contexts
- Manipulate pixmaps (pixel arrays—used to display a graphical object by turning pixels on and off)
- Manage memory associated with graphical window systems
- Handle errors associated with graphical window systems

— Allow inter-visual object communication (via the selection mechanism)

— Initiate other visual object routines (visual object event callbacks)

— Initiate application-specific routines that have been associated with the visual object by the application (application event callbacks)

4.7.4.1.3 Dialog Services

Dialog services provide functions to support high-level graphical window system management for applications with the primary goal of delivering user inputs to the application program and application-driven information to the user. The dialog services allow for a separation of the user interface specifications from the application program. For example, there are many applications that are not concerned with whether a user entry object is a pull-down menu or a scrollable list. These applications are only interested in what the user specified or selected from the user entry object (i.e., the parameter value), which will then trigger some action by the application. To support this notion, a single dialog function might be specified for displaying a window made up of a composite of visual display objects, such as radio buttons, text key-in objects, and scrollable text lists. The application program does not need to manage or understand what the look, location, or visual feedback of any of these items will be. The dialog function has access to the presentation information required to display the specified window and it handles the display of the application specified window. Another dialog service would provide a high-level event loop that returns the user specified input as an application parameter value.

The services provide simplicity over the degree of freedom available in Basic and Toolkit Window Services. Most User Interface Management Systems (UIMS s) provide dialog services to fulfill their requirement of separation of user interface from application software.

These services are subdivided into:

— Window services: provide services used to initialize the window service, create and delete windows with predefined associated visual objects, and manipulation of the pointing cursor. They include services that allow the application to communicate directly with the user via modal or modeless windows.

— Visual object manipulation services: provide services used to access the graphical window system designed by the application designer, display the visual objects defined by the graphical window system, and associate them with application-tied inputs and outputs.

— Event control services: provide services to allow the application to define a set of events and handle triggered events in one of two ways:

- Wait on the occurrence of any event, processing triggered events one at a time from an input queue (event-driven method)

- Attach to each event a function that is automatically executed when the event is triggered (callback method)

4.7.4.2 External Environment Interface Services

These services provide support for the actual elements with which the user physically interacts. These functions provide services in three areas:

- Graphical window system: provides definition of the presentation and behavior of the visual display objects, command language definition (syntax and semantics), specifications related to keyboards, selection devices, audio and video input/output devices.
- Information Interfaces: provides specification of user resource data formats, containing presentation and action information pertaining to visual display objects.
- Network Interfaces: provides protocol services for data transport, which is basically the bottom six layers of the OSI model

4.7.4.3 Interapplication Entity Services

These services provide support for general conventions and specifications for interaction between graphical window system components. The services provide support for generalized network/multisession services, such as message handling between graphical window system components, intermediate language definition, and a standard definition of the format used for saving the presentation, behavior, and action information about graphical window system objects.

4.7.4.4 Windowing Resource Management Services

These services provide general management functions across the graphical window system components, which include system administration-oriented functions (i.e., management of graphical window systems within the scope of the administrator, such as setting up defaults and user customization functions. For instance, it is important to allow reconfiguration and addition of terminals and displays without affecting the application interface.) These resource management services are independent from the OLTP Resource Management Services defined in 4.6.4.3.

A standard definition of the format used for saving the presentation, behavior, and action information about graphical window system objects would provide a vehicle for exchanging graphical window system information between software tools, such as User Interface Management Systems (UIMS s) and Interface Design Tool (ITD s).

4.7.5 Standards, Specifications, and Gaps

Standards that relate to the user reference model presented earlier are considered here. Related standards that might be relevant for one or more of the interface components will also be mentioned.

4.7.5.1 Current Standards

No current international or national standards exist for the graphical window system services, primarily due to the recent emergence of the windowing technology. However, several standard activities are underway and referenced under 4.7.5.2.

Table 4-9 – Windowing Standards

Service	Type	Specification	Subclause	
Basic Window Services	G	X Window System (X-lib)	4.7.5.3	E
	E	ANSI X3K13.6	4.7.5.2	E
Toolkit Window Services	G	X Window System (Xtk)	4.7.5.3	E
	E	ANSI X3K13.6	4.7.5.2	E
	E	IEEE POSIX. 2	4.7.5.2	E
	E	IEEE POSIX .1 {2}	4.7.5.2	E
Dialog Services	G	–	4.7.5.3	E
EEI Services	E	ANSI X3V1.9	4.7.5.2	E
	E	ISO/IEC JTC 1/SC18/WG19	4.7.5.2	E
	E	ANSI HSF -HCI	4.7.5.2	E
	E	ISO TC159/SC4/WG5	4.7.5.2	E
	E	P1201.2	4.7.5.2	E
Interapplication Entity Services	G	X Window System (X protocol)	4.7.5.3	E
Window/Character Resource Management Services	G	–	4.7.5.3	E

4.7.5.2 Emerging Standards

— ANSI X3K13.6. Currently developing an X Protocol standard.

— ANSI X3V1.9. User-System Interfaces and Symbols: Working on a ISO/IEC Standard 9995, Keyboard Layouts for Text and Office Systems. Also working on the Voice Messaging User Interface Forum (VMUIF). This is a mirror standards effort with ISO/IEC JTC 1/SC18/WG19.

— ISO/IEC JTC 1/SC18/WG19. User-System Interfaces and Symbols. Working on developing standards for user interfaces and symbols associated with text and office systems.

— ANSI HFS-HCI. Working on drafts on the design process, information presentation, forms-based dialogs, and window-based interaction.

— ISO TC159/SC4/WG5. Software Ergonomics and Man-Machine Dialog: Working on developing parts of the ISO Standards 9241, Ergonomics of Visual Display Terminals. Their areas of concentration are software ergonomics, dialog principles, dialog styles, methods for evaluating software usability, coding and formatting of information, and terminology

— IEEE P1201. Application and User Portability: Chartered to develop standards that facilitate application and user portability in the X Windows environment. P1201.1 is involved in defining a set of virtual toolkit services that would be independent of any windowing system. P1201.2 is involved in defining drivability guidelines.

— ANSI CODASYL. Working draft available for Forms Interface Management Systems (FIMS), which covers the interface between a programming language and any form fill-in application on a computer or terminal screen.

4.7.5.3 Gaps in Available Standards

There is a de facto standard for the base window system. The X Window System windowing protocol and the Xlib functional interface to the protocol were developed at Massachusetts Institute of Technology. Development is continuing under the aegis of the X Consortium, a group of interested parties in the computer industry and computer manufacturers. Relevant documents from the X Consortium are ''XWindow System Protocol, X Version 11,'' ''Xlib – C language X Interface,'' ''X Toolkit Intrinsics – C Language Interface,'' and ''Bitmap Distribution Format 2.1.''

The X Window System protocol and functional interface are considered to be de facto standards in the base window system area because of their widespread adoption by major computer vendors and industry groups.

Within the government, the National Institute of Standards and Technology (NIST) issues Federal Information Processing Standards (FIPS) that require purchases made by the United States Government to adhere to certain standards. NIST has adopted the X Window System Version 11 Release 3's X Window System protocol, Xlib, Xt Intrinsics, and Bitmap Distribution Format as FIPS 158. This is a noncompulsory (i.e., voluntary) standard.

— Object Definition File Format: There are no standards addressing the format used for describing the ''look and feel'' of graphical window system objects.

— Toolkit Services

— Dialog Services

— Interapplication Entity Services

4.7.6 OSE Cross-Category Services

4.7.6.1 Security

The security aspects of graphical window systems and include: E

- Authentication of person at login
- Authentication of person when a service request is made to a client application
- Provisions for visual labeling of sensitive material
- Option selections available in support of sensitive activities
- Prevention of moving data (cut/past) from a more protected security environment to a less protected environment

4.7.7 Related Standards

Currently, the basic windowing services provide a certain level of graphics functionality, but the existing and proposed graphics standards (e.g., PHIGS, GKS) provide a much more comprehensive solution to graphic support. As the graphics and windowing technologies evolve, this distinction between the windowing and graphics services will continue to be blurred. For instance, proposals are already being developed that provide extensions to the X Window System that support 3-D graphics (i.e., PEX, PHIGS EXtensions), and implementations of GKS are currently available that use the X Window System to create the graphics.

4.7.8 Open Issues

- Audio input/output
- Video input/output
- Security
- Desktop. The Desktop, or graphical windowing shell, is a specification for the graphical window system work surface (i.e., the entire display screen).

  The desktop provides the user with a visual interface to available computer resources. A desktop may be characterized as a visual analog of the POSIX shell. It provides access to system resources, such as devices and files, and provides methods to start applications. Desktops typically also provide a set of often used utilities such as a calendar, a notepad, etc. The desktop is an important component of the look and feel of a graphical window system, but the current state of the industry is too immature for any standardization to materialize on a desktop specification in the immediate future.

  NOTE: There are some valid arguments for defining some requirements for standards at this level. The goal is to enable a user to easily go between application platforms and operate common functions in a similar manner.

4.8 Graphics Services

Responsibility: John Williams

4.8.1 Overview and Rationale

Graphics Services are key components and play an important role in the POSIX Open System Environment as it is used today in many different areas of industry, business, government, education, entertainment, and most recently, the home. The number of applications is growing rapidly, with increasing graphics capabilities. Some of these areas are user interfaces, computer-aided drafting and design, electronic publishing, plotting, simulation, animation, scientific visualization, art, and process control. The use of pictorial graphics provides a more intuitive interface and thus facilitates man/machine interaction.

Graphics has become a routine part of most organizations today, ranging from hardcopy graphs and charts to user interfaces and complex 3-D visualizations incorporating video and sound. The graphics technology of rendering objects has become dramatically more realistic and hence is used by engineers, architects, artists etc., to enable them to see precisely what their final products, whether automobiles or buildings, will look and behave like under real-world conditions.

Graphics has allowed dramatic improvements in the ‘‘look and feel’’ of user interfaces and the trend is towards increasing use of these interfaces to interact with computers graphically, via windows and icons and this reduces the time involved in learning to use a computer.

Standardization of graphics services has many benefits for application developers, users, and systems integrators. The underlying motivations for considering graphics standards and their relation to the POSIX Open System Environment include:

(1) **Portability:** In order to protect investment and achieve independence from a particular technology and a particular supplier of technology, portability at both hardware and software levels is necessary. There are many aspects of portability within graphics, all of which are potential money and time savers.

— Applications portability

— Graphics package portability

— Host machine independence

— Device independence

• input devices: dials, mouse, tablets etc.

• output devices: plotters, raster, vector etc.

— Window system independence

— Programming language independence

— Programmer portability

— User portability

(2) **Interoperability/Distributed Graphics:** In order to allow applications to execute on one machine and display graphics on remote display servers, standard graphics protocols are necessary. This allows for display of graphics on machines that are incapable of executing particular types of applications and it also facilitates graphics conferencing.

(3) **Graphics Data Exchange:** In order to share or exchange graphical information between diverse applications, standard graphics data exchange mechanisms are necessary.

This clause presents a reference model for this component and describes the services provided to application programmers and users. It also describes the current national/international standards, emerging standards, de facto standards, and any existing gaps that need new standardization efforts.

4.8.2 Scope

Included within this component are standards in the graphics area that address the following topics :

— Application Program Interface (API) Standards

— Language Bindings Standards

— Metafile and Archive Standards

— Device Independent Interface/Protocol Standards

— Computer Graphics Reference Model

— Conformance Testing of Implementations of Graphics Standards

— Distributed Graphics Standards

— Imaging Standards

— Performance Metrics Standards

The standards not addressed here are:

— Data Exchange Standards

— Graphical User Interface Standards

— Window Management System Standards

4.8.3 Reference Model

Over the past decade many computer graphics standards have been developed. While they are similar in concepts, their underlying reference models are different. This restricts the degree to which the standards are compatible. By producing a reference model to which all future graphics standards are to adhere, compatibility of graphics standards is assured.

Formal work on the Computer Graphics Reference Model (CGRM) standard is in progress within the ANSI X3H3.2 committee. It is an international standard that explains the relationships between existing graphics standards and defines relationships between standards in computer graphics and those in other areas. It will form the basis for the next generation of computer graphics standards. Broadly speaking, CGRM provides a framework within which relationships between standards can be described.

There are five types of standards in the current family:

— *Application Program Interface (API) Standards:* These define a programming interface for application programmers. GKS, GKS-3D, PHIGS, and Xlib are examples of standards in this area.

— *Metafile and Archive Standards:* These standards define representations of graphics for storage and transfer between systems. These are basically file format and file transfer encoding standards. CGM (Computer Graphics Metafile) and PHIGS Archive files are of this type.

— *Device Independent Interface Standards:* These standards define the interface between device-independent graphics systems software and one or more device-dependent graphics device drivers. CGI (Computer Graphics Interface) is the standard in this area.

— *Language Binding Standards:* API and device interface standards are functional specifications defined independently from particular programming languages. Each standard has attached language binding standards that state how the functionality should be accessed from a variety of programming languages.

— *Framework Standards:* These include the standardization of a reference model for computer graphics, conformance criteria, and the registration of graphical items.

The CGRM describes the current family of graphics standards in terms of the following four levels of abstraction:

— *Application Level:* This is the level at which applications-related information is composed into abstract graphics related to the application.

— *Virtual Level:* At this level, the graphical output to be displayed is described in terms of output primitives

— *Logical Level:* At this level, the information necessary to render a primitive on a particular device is assembled.

— *Physical Level:* This level is associated with a particular output device and a collection of input devices. The physical level need not correspond to real devices such as a pen plotter. There could be further layers of the system between the physical level and the hardware, such as the window system.

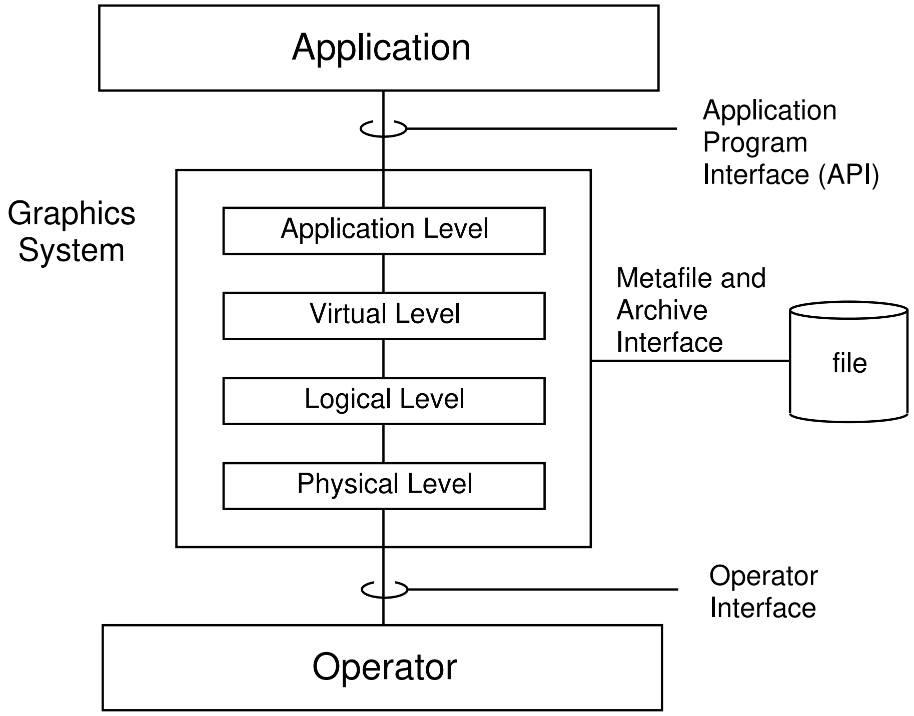

Figure 4-16 – Computer Graphics Reference Model Level Structure

The Application Program Interface (API) is the interface between the application and the graphics system. There are also interfaces to metafiles and archives and to the operator. Here the operator need not mean human operator, but the user of the graphics system; for example, the window system.

The Computer Graphics Reference Model can be incorporated into the POSIX OSE reference model as depicted in Figure 4-17. It provides the context for the discussion of graphics services and shows that the graphics services is a component of the overall POSIX OSE and is available to the the application through the POSIX OSE API.

The entities and interfaces specific to the graphics services are identified in this clause.

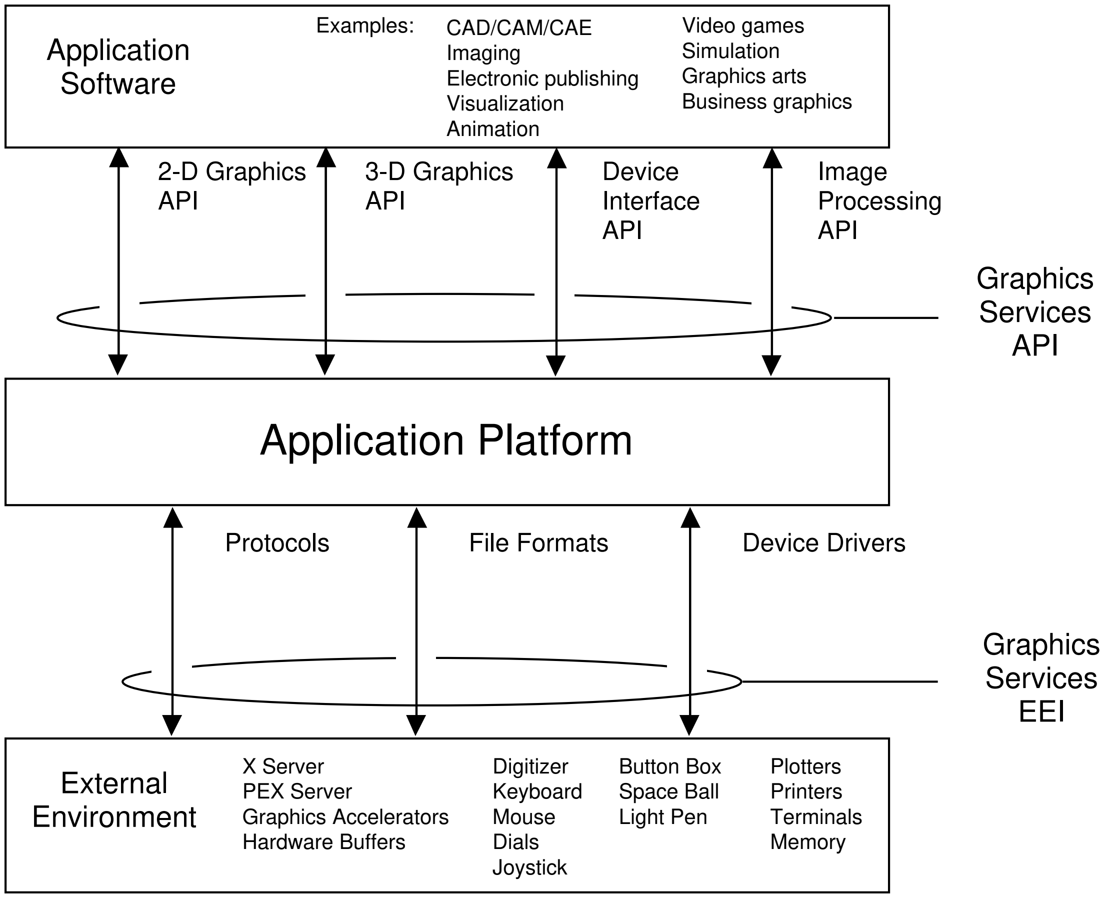

Figure 4-17 – POSIX OSE Graphics Service Reference Model

The entities are:

(1) **Application Software**, such as CAD/CAM/CAE applications, imaging applications, electronic publishing, etc.

(2) **Application Platform**, which consists of graphics libraries such as GKS, PHIGS and Xlib.

(3) **External Environment**, consisting of external entities with which the application platform exchanges information such as input devices, X/PEX servers, hardware buffers, etc.

The interfaces are:

(1) **Application Program Interface (API)**, which is the programming interface between the application and the application platform. It standardizes the conceptual model, calling sequence, functions, and syntax that a programmer uses to develop a graphics application. Each API standard has an attached language-binding standard that allows the

functionality to be accessed from a variety of programming languages. A standard API in conjunction with a standard language binding promotes application portability, by isolating the programmer from most hardware peculiarities and providing language features readily implemented on a broad range of processors. Examples of APIs in the graphics services area are GKS, PHIGS, PIK, PostScript, etc.

(2) **External Environment Interface (EEI)**, which is the interface between the application platform and the External Environment described earlier. In the graphics services area these can be device drivers that are used for communication between the device-independent and the device-dependent functions as well as protocols and file formats.

The standardization efforts in the graphics area focus on these two interfaces.

4.8.4 Service Requirements

4.8.4.1 Graphics Concepts

Computer Graphics Services can be discussed in terms of the following fundamental graphics concepts:

Output Primitives

The output primitives are the building blocks used to construct graphical objects for display or storage in an archive file. Common output primitives are:

— *Line*

— *Polyline* used to represent a series of straight lines from a set of points.

— *Marker* is a special symbol used to represent semantics of graphical objects.

— *Fill area* is an area with an edge and an interior which may be filled with a solid color or some form of pattern or hash.

— *Text* is an output primitive used to represent strings in two or three dimensional space.

— *Annotation text* is text that is always displayed facing the viewer.

— *Cell arrays* are areas with rectangular grids which can take on individual colors.

— *Triangle strip* is a set of triangles defined by a particular ordering of vertices.

— *Quadrilateral mesh* is a set of quadrilaterals defined by a grid of vertices.

— *Surfaces:* NURBS (Nonuniform Rational B-Spline)

— *Curves:* NURBS (Nonuniform Rational B-Spline)

— *Conics:* Circles, ellipses, parabolas, and hyperbolas

Primitive Attributes

Attributes of primitives determine the style of the display of the primitive. For example, lines and edges may have different line styles such as dotted or dashed, text may have different fonts, orientation, and character spacing. A polymarker may be an asterisk or a small triangle. They all may be red in color. General type attributes that apply to almost any output primitive are color, visibility, pickability, and highlight method.

Input Primitives

Input primitives or logical devices are virtual devices designed to insulate the application from the real input devices. Logical devices include picking devices, locator devices, choice devices, valuator devices, etc. In terms, of actual devices, a locator device might be associated with the first mouse button.

Input Model

The input model describes how input primitives and logical devices are related to physical input devices and the degree of control provided to the application over the devices. For example, one control choice might be how feedback is echoed to the operator when a logical locator device is attached to a depressed mouse button. The feedback might be a rectangular cursor or the highlighting of geometry as a cross-hair cursor moves over it. When the button is released the device coordinates are placed in the locator data record and an event is placed in an event queue for which the application can check asynchronously. The method the application uses to determine if a device has data for it is usually described in terms of modes. A common mode is event mode. The application waits a finite time for some event to appear in a queue. If no event comes in the finite time, the application does other processing and eventually comes back to check the queue with the wait for some event. If an event appears, the application determines what type it is and gets the data for that type of event. For a pick device, the data might be all possible graphical primitives that could intersect some aperture, possibly specified in the device coordinate system.

Coordinate Systems and Clipping

Part of the graphics services is a means to utilize various coordinate systems. Graphical output has to be described to the graphics system in terms of some coordinate system, relevant to the application and presented to the display device in terms of its own coordinate system, the device coordinate system. It is unlikely that these two coordinate systems will be the same. A graphics system may therefore involve a number of coordinate systems and hence the need to define transformations between them. Some standard types of transformations are scaling, rotating, translating, reflecting, and projection, such as parallel and perspective. They are used to manipulate objects in a coordinate system and to map from one coordinate system to another. The coordinate systems commonly used are modeling coordinates, world coordinates, view-reference coordinates, normalized projection coordinates, and device coordinates.

Clipping is the process of specifying a region in space and restricting graphical output to that region. Only those primitives that define objects in that region will have their output displayed.

Output Model

The output model is the concept of how graphics objects are created, displayed, and controlled on output devices. The output model defines how to position and organize objects on the screen, and the visual state of these objects such as visible or invisible, hidden lines removed or not removed, picture matches retained structure, picture not consistent with retained structure, etc.

More specifically, the output model concept is made up of the:

— Transformation pipeline

— Rendering pipeline

— Retained structures

— Nonretained structures

— Graphics state

— Window systems

Storage/Archiving

Storage data formats for displayed or rendered images are required, but not treated at this time.

4.8.4.2 Graphics Requirements

The graphics service requirements of all users of this system can be generalized as:

— The ability to create, delete, and modify output primitives.

— The ability to specify and edit the primitive attributes globally and individually.

— The ability to transform (i.e., scale, translate, rotate, reflect, project, etc.) primitives for construction of more complex objects and for arrangement in the viewing space.

— The ability to create and manipulate a database of primitives, to define and edit attributes, to create and combine transformations, and to selectively control the display of graphics primitives.

— The ability to display graphical objects constructed in a retained database, or the ability to display primitives immediately, or to display from both a retained database and immediately.

— The ability to apply lighting and shading algorithms to collections of graphical objects with multiple light types and sources.

— The ability to prepare display data and control the timing of the actual display of the display data. On some systems this is referred to as frame buffer control.

— The ability to store and retrieve graphical objects from files.

— The ability to control input devices and retrieve data from input devices.

— The ability to direct output to a meta-file and retrieve graphics data from a meta-file.

— The ability to inquire about all aspects of the graphics environment; e.g., the state of the system at any given time, the actual capabilities of a given hardware platform, the attributes and primitives supported by a given implementation, etc.

— The ability to distribute graphics.

— The ability to control errors.

4.8.4.3 Application Program Interface Services

The major categories of graphics services available in the POSIX OSE API area include:

— 2-D graphics API services

— 3-D graphics API services

— Device interface API services

— Image processing API services

For most of these API standards there exist standard language bindings so that applications using different programming languages can access the same functionality.

The choice of which graphics standard API to use will depend on a number of factors: application profile, overall system architecture, equipment available, existing application database interaction, system performance considerations, user interface requirements, management policy, and other external factors. The aim of producing a compatible set of graphics standards in GKS, GKS-3D, PHIGS, PHIGS PLUS, etc. (described in the Standards subclause) is to allow that choice to be made in the most flexible way.

4.8.4.4 External Environment Interface Services

The major categories of graphics services in the POSIX OSE EEI area include:

— Protocols

— File Formats

— Device Drivers

The choice of which standard to use depends on a number of factors: application profile, system architecture, equipment available, system performance considerations, and other factors

E

4.8.5 Standards, Specifications, and Gaps

There are several major standards existing in the computer graphics industry today, that have been approved by National/International organizations such as ANSI, ISO, and IEEE. There are also standards efforts going on in related areas such as application data exchange. These official graphics standards are complemented by de facto standards that have been accepted by the graphics industry at large. This document provides a general explanation of these standards, their specifications, and interrelationships.

4.8.5.1 Current Standards

PHIGS — ISO 9592 Parts 1–3
Fortran Language Binding — ISO 9593-1
Ada Language Binding — ISO 9593-3
C Language Binding — DIS 9593-4

The Programmer's Hierarchical Interactive Graphics Standard (PHIGS) is a functional specification of the interface between an application program and its graphics support system. It is an ANSI/ISO standard and provides the following graphics functionality:

— A high degree of interactivity

— Multilevel, hierarchical structuring of graphics data

— Easy modification of graphics data and the relationships among the data

— 3-D, as well as 2-D, graphical input and output

— Offline storage (and retrieval) of graphics data

PHIGS controls the definition, modification, and display of hierarchical graphics data and specifies functional descriptions of systems capabilities, including the definition of internal data structures, editing capabilities, display operations, and device control functions. PHIGS manages the organization and display of data in a centralized database, allowing programmers to define and organize graphical data in a manner most convenient to the application. Such a hierarchical approach is a big benefit and is not available in GKS, another international standard.

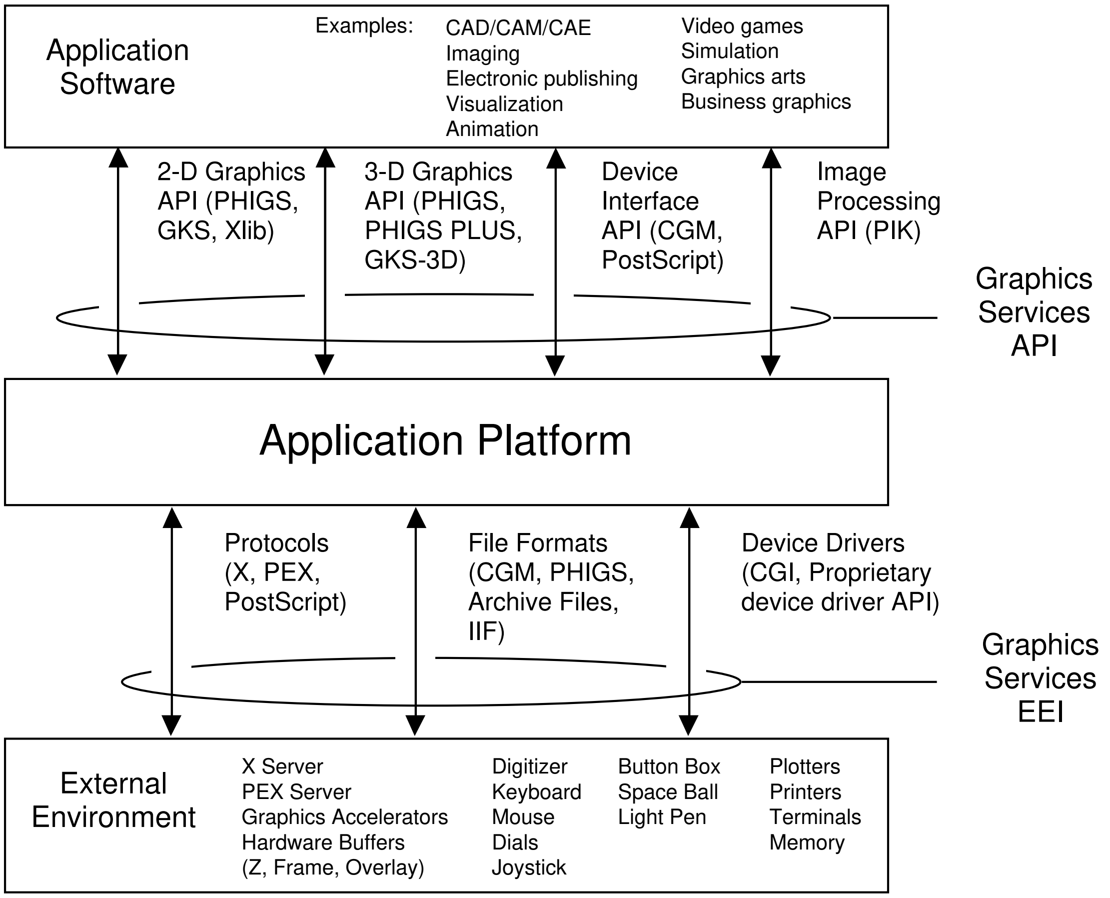

Figure 4-18 – POSIX OSE Graphics Service Reference Model Standards

Objects are defined in the PHIGS graphical database by a sequence of elements, including output primitives, attributes, transformations, and invocations of other object and object part definitions. These elements are grouped into entities called structures. Structures may be related in a number of ways, including geometrically, hierarchically, or according to inherent properties or characteristics, as defined by an application.

PHIGS provides tools to use hierarchical data structures with minimal effort by the application programmer. Pictures constructed from geometric models often have a clearly evident structure. This structure can sometimes be easily seen in the repeated use of symbols, in the connections and geometric relationships between objects, or in the overall organization of a complex image. Even if the object's structure is not evident, its underlying data organization may be quite rigorous, well defined, and well understood by the application. PHIGS supports both these cases by separating the definition of graphics data from the actions required to display them.

Table 4-10 – Graphics Standards

Service	Type	Specification	Subclause
PHIGS	S	ISO 9592-1, -2, -3	4.8.5.1
PHIGS PLUS	E	ISO DIS 9592-4	4.8.5.2
GKS	S	ISO 7942	4.8.5.1
GKS-3D	S	ISO 8805	4.8.5.1
CGI	E	ISO DIS 9636	4.8.5.2
CGM	S	ISO 8632-1, -2, -3, -4	4.8.5.1
PHIGS Archive files	S	ISO 9592-2, -3	4.8.5.1
IPI	E	JTC 1 N1002	4.8.5.2
Conformance Testing	E	ISO DIS 10641	4.8.5.2
PEX	G	MIT Consortium	4.8.5.3
Graphics Style Guide	G	–	4.8.5.3
Control and Deterministic Functionality	G	–	4.8.5.3
CGRM and Windows	G	–	4.8.5.3
Solids	G	–	4.8.5.3
Cut and Paste	G	–	4.8.5.3
Nonretained Graphics	G	–	4.8.5.3

Table 4-11 – Graphics Standards Language Bindings

Standard	LIS	Ada	APL	BASIC	C	C++
PHIGS		S			E	
GKS		E			E	
GKS-3D		E			E	
CGI					E	

Standard	COBOL	C-LISP	Fortran	Pascal	PL/1	Prolog
PHIGS			S			
GKS			S	S		
GKS-3D			E	E		
CGI			E			

NOTES: LIS — Language-independent specification is available.
Ada, APL, BASIC, — Language-dependent specifications exist.
S, E, G — Standard, Emerging Standard, Gap

The structured definition of graphics data inherently reduces repetition and connectivity problems. The repeated use of component objects and the relationships between them can automatically be made a part of an object's definition.

The structured definition of data allows images to share component objects, making it faster and easier for application programs to define and modify picture descriptions. Sharing component objects will also reduce storage requirements for graphics data.

PHIGS permits rapid dynamic access to a centralized graphics database. This allows PHIGS to support interactive end user application programs and, depending on the capability of the hardware, realtime definition, and modification of graphics data. PHIGS is capable of performing three-dimensional modeling transformations, workstation transformations, and viewing. It also handles two dimensions through a shorthand functionality of three dimensions. In workstation transformations, PHIGS provides another level of display control after the viewing operation that can isolate a section of an image for pan and zoom operations.

The National Institute of Standards and Technology (NIST) has developed a test system to help determine whether implementations of PHIGS conform to the specifications of the ANSI standard X3.144. The PHIGS Validation Test (PVT) suite consists of highly portable Fortran programs which examine test conditions and report the results.

PHIGS PLUS — DIS 9592-4

PHIGS Plus Lumiere Und Surfaces (PLUS) specifies a set of extensions to PHIGS that addresses some of the deficiencies in the graphics functionality provided by PHIGS. PHIGS does not include ‘‘higher level’’ primitives such as curves and surfaces, and techniques for lighting and shading. Recognizing this, an ad hoc working group was formed to propose a set of extensions to PHIGS to enable these capabilities to be addressed in a standard manner, compatible with the overall philosophy of PHIGS. This set of proposed extensions was submitted to ISO and has since been developed into PHIGS PLUS. PHIGS PLUS enhances PHIGS by providing:

— Primitives for defining curves and surfaces

— Lighting models

— Shading of surfaces

— Depth cueing

— Color mapping and direct color specification

PHIGS PLUS is not an international standard yet and is currently at the stage of committee draft.

GKS — ISO 7942; FIPS 120
Fortran Language Bindings — ISO 8651-1
Pascal Language Bindings — ISO 8651-2
Ada Language Bindings — DIS 8651-3
C Language Bindings — DIS 8651-4

GKS Information Bulletin

The Graphical Kernel System (GKS) is a 2-D graphics system and provides no support for 3-D. It is a 2-D graphics API that shields the programmer from differences among various computers and graphic devices. It allows for portability of graphics applications by standardizing the basic graphic functions and the method and syntax for accessing these functions.

GKS is an ANSI, ISO standard and is widely used today. It has standard language bindings for Fortran and Pascal. Language bindings for C, Ada, and LISP are currently being worked on.

GKS supports the grouping of logically related primitives such as lines, polygons, strings, and their attributes into collections called segments, which cannot be nested.

GKS supports many graphical input and output devices such as black/white and color displays, printers, plotters, mice, data tablets, joysticks, and digitizers.

GKS-3D — ISO 8805
Fortran Language Bindings — DIS 8806-1
Pascal Language Bindings — CD 8806-2
Ada Language Bindings — DIS 8806-3
C Language Bindings — DIS 8806-4

Graphical Kernel System for Three Dimensions (GKS-3D) is an ISO standard and specifies extensions to GKS for defining and viewing three-dimensional wire-frame objects. In addition, the GKS input model has been extended to provide three-dimensional locator and stroke input. GKS-3D allows the operator to obtain information from three-dimensional input devices and to perform hidden line/hidden surface removal (HLHSR) at the workstation. It does not, however, provide specific functions for controlling rendering techniques such as light source, shading, texturing, and shadow computations that must be done locally at the workstation. Conceptually, all workstations are three-dimensional in GKS-3D, which is made possible by shielding the hardware peculiarities as in GKS.

CGI — DIS 9636 Parts 1–6
Fortran Language Bindings — DIS 9638-1
C Language Bindings — CD 9638-4

The Computer Graphics Interface (CGI) specifies a standard functional and syntactical specification of the control and data exchange between device-independent graphics software and one or more device-dependent graphics device drivers. Unlike the graphics standards discussed earlier, CGI specifies an interface at the device-driver level, rather than at the application level.

Unlike CGM, which only handles graphical output, CGI handles both input and output, which makes all devices appear as identical, virtual graphics devices. Therefore, this protocol is also known as the Virtual Device Interface (VDI). It provides a standard graphics escape mechanism to access nonstandard graphics device capabilities. CGI allows programmers to write portable device-driver software that is independent of the physical graphics device characteristics. This makes the software portable and compatible with a wide variety of devices.

CGM — ISO 8632 Parts 1–4

The Computer Graphics Metafile for storage and transfer of picture description information (CGM) is a mechanism for retaining and/or transporting graphics data and control information. This information contains a device-independent description of a picture at the level of the Computer Graphics Virtual Device Interface described above. It provides a standard graphics escape mechanism to access nonstandard graphics device capabilities via the metafile.

Pictures are described in CGM as a collection of elements of different kinds, representing, for example, primitives, attributes, and control information. It is multipart ANSI, ISO standard. Part 1 contains the semantics of all the elements. Parts 2, 3, and 4 contain the syntax of three different bindings of the standard, namely: character-coded, binary, and clear-text encodings.

PHIGS Archive files — ISO 9592 Parts 2–3

Parts 2 and 3 of the PHIGS standard define an archive file format for storage and transfer of PHIGS structures and structure network definitions from the CSS (Central Structure Store). Part 2 describes the file format and Part 3 a clear text encoding. This encoding is constructed using the same techniques as used by CGM.

4.8.5.2 Emerging Standards

IPI — JTC 1# 1002

Image Processing and Interchange is a functional specification and several language bindings for an Application Programmer Interface to Imaging. The standard defines the data objects, primitive operations, and a reference model. The API supplies the basic building blocks upon which applications requiring imaging functionality can be built within conventional, distributed, and image oriented computing environments.

The International Standard for Image Processing and Interchange includes three parts:

Part 1 Common Imaging Architecture

Part 2 Programmer's Imaging Kernel (PIK)

Part 3 Image Interchange Format

Conformance Testing of Implementations of Graphics Standards — DIS 10641

The existence of any standard brings up the question of how one can be sure whether a product claiming to conform to the standard does in fact conform. If this question is not addressed then the process of standardization becomes pointless.

The general approach to software validation is through testing. The method is to subject the software to a collection of test cases and observe the results. If the results are different from what is expected, the software does not conform to the specification. The ANSI X3H3.7 committee is working on a standard that specifies the characteristics of standardized test sets for use in determining the conformance of implementations of graphics standards. It will also provide guidance to functional standards developers concerning the content of their standards and the conformance rules within standards.

4.8.5.3 Gaps in Available Standards

4.8.5.3.1 Public Specifications

PEX — PHIGS Extensions to X

PEX is a network protocol extension to the X Window System. As many applications require 3-D graphics and other forms of input devices such as dials and button boxes, all of which are not supported by X, it became necessary to extend the X Protocol to include 3-D graphics. PHIGS was selected as the application program interface because of its acceptance as a 3-D standard, its high degree of input ability, and its powerful database editing capabilities. In 1988, the MIT X Consortium contracted to add 3-D and extended input extensions to the X protocol and the first release of PEX as a sample implementation (PEX-SI) was made in January 1991 but is not yet available commercially. Using PEX, PHIGS workstations would be defined as X Windows. For the programmer, X, PHIGS, and PEX standards provide portability.

4.8.5.3.2 Unsatisfied Service Requirements

- Applications have different behaviors for similar functions which hinders user portability. By adopting a uniform approach (Graphics_Style_Guide) users can switch between applications without a lot of training.
- Current existing standards allow a wide interpretation for implementors of the standards thus denying the applications useful controls. In order to achieve true portability in a distributed environment, applications will need control and deterministic functionality.
- How window standard fits into CGRM
- Current existing standards do not address solids.
- The ability in a standard defined way to perform cut and paste between applications.

— Current standards do not allow nonretained graphic methods to do lighting and shading.

4.8.6 OSE Cross-Category Services

Not applicable.

4.8.7 Related Standards

IGES , NBSIR 86-3359

See 4.5.

X Window System Data Stream Definition Parts 1-4

(Being worked on in ANSI X3H3.6)

Part 1: Functional specification
Part 2: Data Stream Encoding
Part 3: KEYSYM Encoding
Part 4: Mapping onto Open Systems Interconnection (OSI) Services

The X Window System is a network based windowing and 2-D graphics system. It uses the client-server model. The client and server can reside on the same or different platforms. The client is an application program executing anywhere on the network and displaying on the screen. It does this by making calls to a library called Xlib to generate protocols. The X server is the software that accepts protocols sent by the client and processes them for display. It also accepts input from a mouse or keyboard for return to the application program. The X protocol specifies the data stream encoding between the server and the clients. The X Protocol originally developed by the X Consortium at MIT, is being standardized by the ANSI X3H3.6 committee. The encoding will provide a standard interface for applications running on both distributed and nondistributed environments having high-speed, reliable, network based communications.

X Protocol is designed to work in a heterogeneous network environment. Below the X Protocol, any lower layer of network can be used, as long as it is bidirectional. Currently TCP/IP and DECnet are the two network protocols commonly supported in X servers. Part 4 of this standard specifies the mapping of X Windows onto the OSI Services.

XLIB

Xlib—C Language X Interface is the common component of X Windows and resides on all X-based systems. Although X is fundamentally defined by a network protocol, application programmers do not interface directly with the X Protocol. Instead, they interface to the X Protocol through Xlib.

The X Window System uses the client-server model. The client is an application program executing anywhere on the network and displaying on the screen. It does this by making calls to Xlib to generate protocols. The X server is the software that accepts protocols sent by the client and processes them for display.

From a graphics perspective, Xlib is a 2-D graphics library and provides graphics primitives like points, lines, and arcs. It has a Graphics Context (GC) to allow modification of graphics attributes such as line type, line width, color, and font type. The Xlib developed initially at MIT is in the Public Domain and is a de facto standard for windowing and 2-D graphics. It has been adopted by major computer vendors and industry groups. It is currently being considered for standardization by the IEEE P1201 committee.

PostScript

The PostScript language from Adobe Systems Incorporated is a simple interpretative programming language with powerful graphics capabilities that has become a de facto industry standard. It is a high-level, device independent language that is primarily used to describe the appearance of text, graphical shapes, and images on printed pages or screens. Programs written in this language may be used to communicate information from a composition system to a printing system. PostScript programs are created, transmitted, and interpreted in the form of source text and there is no compiled or encoded form of this language.

SGML, ISO 8879: 1986

See 4.5.

IGES/PDES Organization (IPO)

See 4.5.

ISO/IEC TC184/SC4 (STEP)

See 4.5.

ISO/IEC TC130 (Color Prepress)

ISO/IEC JTC 1/SC18 (Text and Office Systems

ISO/IEC JTC 1/SC29 (Multimedia Coding)

E

4.9 Character-Based User Interface Services

Responsibility: Martial Van Neste

4.9.1 Overview and Rationale

This clause describes the system services that are related to character-based terminals. It describes both the application program interfaces to character-based terminals and also the look and feel of the interaction between the user and the user interface equipment.

Despite the attention paid to graphical window interfaces, the vast majority of applications are written with a character based user interface. In fact, character-based devices are best suited for applications where the constraints of cost, speed, and the clutter of a pointing device on the desk are a major concern.

It should be noted also that there are character-based window applications that may not have all the flexibility and ease of use of their graphic counterparts, but represent an alternative allowing the utilization of the large installed base of character terminals and still improve the ease of use.

This clause is one portion of the User Interface API and EEI as described in Section 3.

4.9.2 Scope

The scope of this clause is limited to the services and standards required to support character (non-bitmapped) terminals. The services described here do not preclude the use of block-mode terminals, even though most applications built on POSIX -compliant platforms historically have used character-stream terminals.

4.9.3 Reference Model

This subclause identifies the entities and interfaces specific to the character-based terminal services of an OSE.

As illustrated in Figure 4-19, the components of character-based interfaces are broken into two groups: those specifications that impact the application programming interface and those that impact the external user interface.

This reference model is consistent with and expands on the reference model in Section 3.

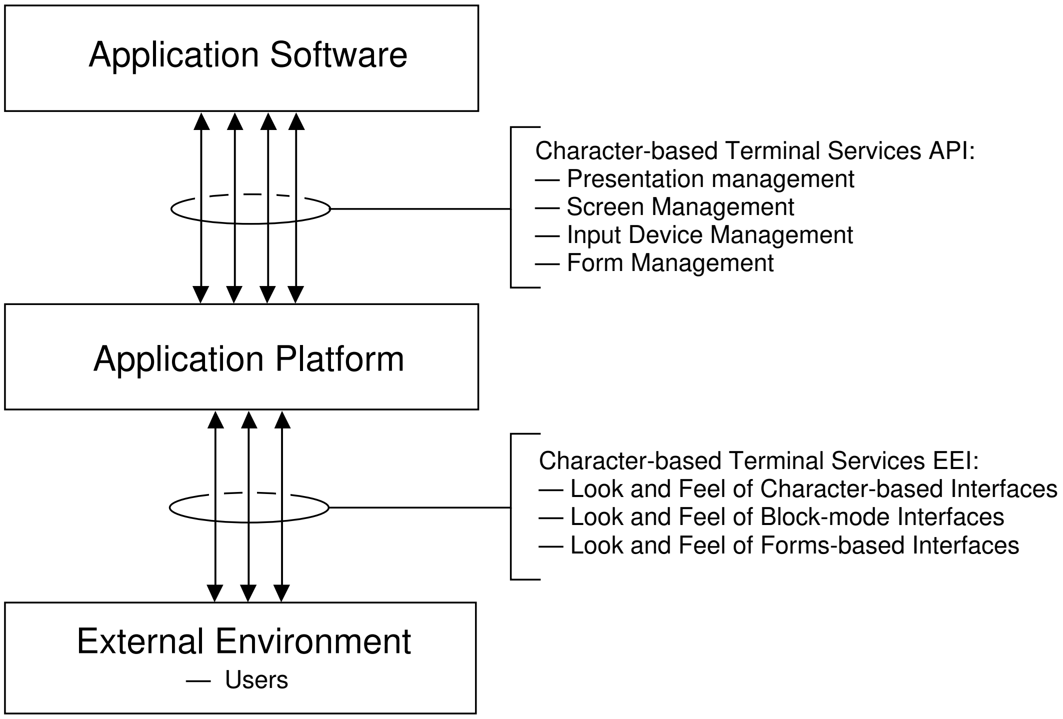

Figure 4-19 – Character-based Terminal Reference Model

4.9.4 Service Requirements

The fundamental service requirements for character-based terminals are to allow applications to be written that make use of the features of a wide variety of terminals using a single terminal-independent interface. The look and feel of user interactions should be consistent between applications to make moving between applications as simple as possible.

4.9.4.1 Application Program Interface Services

Application services include those made available to the application developer to separate the application function from the user interface functions as much as possible.

These standard services support requirements for application portability and terminal independence.

Presentation Management

Functions available for Presentation Management are:

— Placement of text on the screen using a consistent reference

— Positioning of the cursor for further output on the scree or for user input

— Control of attributes of displayed text such as highlighting, underscoring, and coloring, if available

— Clearing or refreshing the screen

— Getting the current cursor position

Screen Management

Functions available for Screen Management are:

— Control of the number and the width of the lines displayed

— Use of a protected status line

— Protection from writing or clearing in defined portions of the screen

— Auto-wrapping in defined portions of the screen

Input Device Management

Functions available for Input Device Management are:

— Configuration of the function keys, if available

— Keyboard locking

— Changing key mappings

Form Management

Functions available for Form Management are:

— Definition of a form with different output and input text fields

— Definition of the attribute input fields, such as text or different numeric formats

— Generic and customizable error handling procedures for incorrect input

4.9.4.2 External Environment Interface Services

The look and feel of user interactions with applications should be standardized to make moving between applications as simple as possible. The areas that require standardization are:

— Style of selecting commands

— Accessing online help

— Performing common functions such as page forward and page backwards.

— Selecting or moving between fields in a forms-based environment E

These interactions will differ slightly between different types of terminals because of limitations of the terminals.

4.9.4.3 Related Service Requirements

To be provided.

4.9.5 Standards, Specifications, and Gaps

4.9.5.1 Current Standards

None.

4.9.5.2 Emerging Standards

FIMS

ANSI CODASYL. A working draft is available for Forms Interface Management System (FIMS), which covers the interface between a programming language and any form-filling application on a computer or terminal screen.

This specification addresses some of the services requirements for a forms-based user interface.

4.9.5.3 Gaps in Available Standards

4.9.5.3.1 Public Specifications

E

Curses

Curses is a set of subroutines that provide a terminal-independent interface to applications. Many different types of character-based terminals are supported. Curses lacks complete support for flexible user input.

This specification satisfies some of the service requirements for character mode terminals. A recent specification for Curses can be found in volume 3 of X/Open's XPG 3. E

4.9.6 OSE Cross-Category Services

4.9.6.1 Security

To Be Provided.

4.9.6.2 Administration

It is important to allow the system management personnel to configure the system to designate where each terminal is connected. Also needed is the ability to add support for new terminals without affecting the application interface.

4.9.6.3 Configuration Management

The system could include a descriptive database of a current set of supported terminals, so that terminal-independent services can do the mapping for the different functions.

4.9.7 Related Standards

None.

4.10 User Command Interface Services

Responsibility: Wendy Rauch

4.10.1 Rationale and Overview

Although system-level services are necessary for application portability and interoperability, they are insufficient for many users' system needs. To maximize portability, users also require the commands, command interpreter (shell), compilers, editors, and other utilities that have been traditionally associated with many operating systems. These command interface services facilitate a successful port and help users to manage and maintain applications and to solve problems on an ad hoc basis. The standardization of these utilities allows users and programmers to move from platform to platform without having to relearn the command interface for each application platform.

4.10.2 Scope

This clause describes how a user interacts with an application platform by executing general purpose commands. This command interface is also available to applications so that applications also can execute commands. A standardized command interface provides a consistent, interactive environment across platforms for users and programmers.

Commands that are outside the scope of this clause are:

— System administration and installation commands

— Text formatting programs

— Database commands

— Networking and communications commands

— Graphical user interfaces

Networking commands and graphical user interfaces are described in other clauses of this guide.

4.10.3 Reference Model

The use of the command interface services presented in this clause is consistent with the reference model in Section 3. The POSIX OSE reference model for the command interface also is consistent with typical implementations for user command languages in traditional UNIX-based systems.

As Figure 4-20 shows, the command interface is available both to users (through the External Environment Interface) and to applications (through the Application Programming Interface). Any operating system implementation can reside underneath the APIs and EEIs.

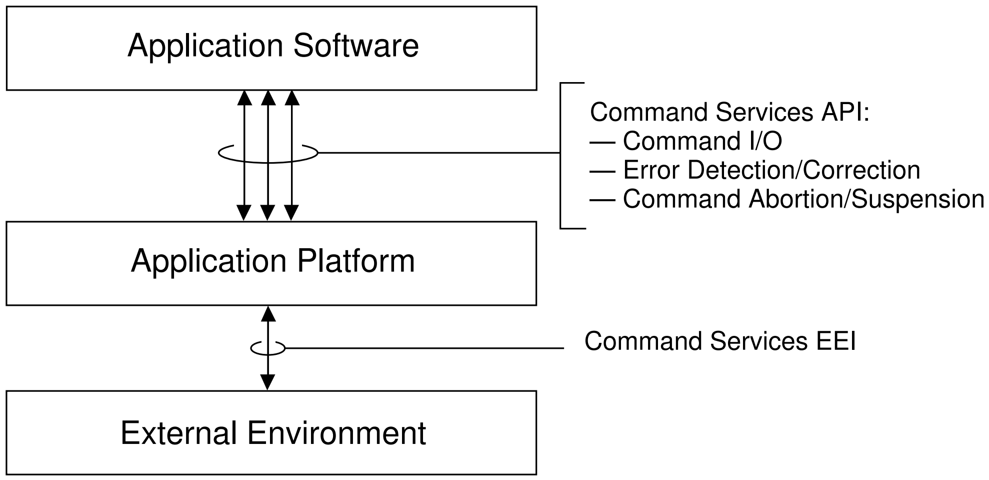

Figure 4-20 – POSIX OSE Reference Model for Command Interfaces

The API and EEI command interfaces provide access to a software component (known as a command interpreter or shell) that interprets the commands issued by either the user or the application. The command interpreter acts as an intermediary between the command API and EEI and the base application platform's system-level services. The command interpreter reads the commands entered and parses them. Depending on the type of command (e.g., utility or built-in shell command), the command interpreter either executes the command for the user or application, using the base application platform's system-level services, or it calls on the system-level services to create a new process which executes the command.

None of the methods of executing commands have an impact on the API or EEI specifications.

The commands interfaces may be available to users and applications either locally or remotely. Remote invocation of a system's command interfaces is provided through networking and data interchange capabilities. These are described in 4.3 and 4.5. Alternatively, remote access to a system's command interfaces may be available through certain interapplication services.

4.10.4 Service Requirements

There are three major aspects of command interface services that must be addressed for practical support of multivendor application portability and system interoperability. The first aspect consists of the basic functionality and interfaces provided for generally usefulness. The second aspect of command interface services concerns the ability to move applications, such as script files, between platforms. The third aspect concerns user portability so that the same user interface is available on different platforms.

Since most command interfaces are available at the API and EEI, the service requirements for the API and the EEI are very similar. This clause, therefore, discusses primarily the EEI command interface requirements. The API service subclause discusses only the additional service requirements for applications.

4.10.4.1 Application Program Interface Services

In a command API, the output syntax of the commands and command responses (such as error messages) need to be standardized, in addition to the calling sequence and allowable inputs. Such standardization is necessary to allow applications executing a command to reliably parse the output of that command.

The API should be able to access all of the services available to the user at the EEI. The additional service requirements for the API are as follows:

— Ability to provide the input to the command and access the output of the command when necessary

— Ability for the application to detect and correct errors as the command is executed

— Ability to abort or suspend the command as it is executing.

It is also important to have the ability to create script files which are combinations of commands. The scripting language developed for this purpose is an application development language. The scripting language has the following requirements:

— Conditional execution primitives

— Repeated execution primitives

— Ability to display output

— Ability to prompt the user for input

— Ability to execute commands and obtain error information.

The services and standards for the scripting language are described in this clause, rather than in the Languages clause 4.1, because it is so closely related to the command interface.

4.10.4.2 External Environment Interface Services

Users need a number of capabilities in order to work on a system. On a traditional system, these are implemented by providing interactive commands entered via a keyboard. However, as graphical user interfaces evolve, these commands may also be implemented by clicking on a mouse in a particular area of the screen, by a touch screen, a tablet, or other input device. E

The major services at the EEI provide the following abilities:

— Capture the output of a command or application into a file

— Redirect the input for a command from a file

— Direct the output of a command to be used as the input to another command

— Execute applications

— Get online help for commands or applications

— Manipulate file contents:

  - Cutting
  - Pasting
  - Concatenating
  - Converting
  - Sorting
  - Reformatting
  - Comparing
  - Searching for regular expression

— Edit files

  - Interactive editors
  - Batch or ''stream'' editors

— Display files

  - Pausing when necessary
  - Display only selected ranges of files

— Manipulate files

  - Create
  - Delete
  - Rename
  - Move
  - Copy

— Print files

— Perform network functions

  - File transfer
  - Remote execution of commands
  - Remote file printing

— Perform batch processing E

  - Create and manage batch queues E
  - Submit, terminate, and get status of jobs E
  - Retrieve output E
- Manipulate and display directories
  - Create
  - Delete
  - Display
  - Destroy (Delete a directory and all its subdirectories and files)
- Control file and directory permissions
- Communicate with other users
  - Electronic mail
  - Online interaction where two or more users communicate with each other simultaneously
- Control the application execution environment
  - Execute applications in the background
  - Abort applications running in the foreground or background
  - Suspend an application
  - Move an application running in foreground mode to the background
- Schedule commands for periodic execution
- Control the users' input equipment, such as a terminal or graphical user interface
- Manage local environment and configuration information
- Query local environment and configuration data
- Configure an environment for an international locale.

E

These services enable remote users and applications to access and execute a system's command interfaces as if they were directly connected to that system. The major categories of interapplication entity services include the following:

- Login and use hosts on a network as if the users logging-in were directly connected to the local terminal
- Remotely execute a system's shell commands as if the user were directly connected to a local terminal
- Copy files between hosts without going through a network file transfer program

— Find out who else is logged into the machines on a local-area network

— Query the status and uptime of all machines on a local-area network.

4.10.5 Standards, Specifications, and Gaps

There are currently no formal standards for command interfaces. There are, however, several command-interface standards-development activities underway. In addition, there are several consortia-defined specifications and de facto specification standards for commands, shell, and utilities services and interfaces.

Table 4-12 summarizes the shell and utilities standards and specifications and work in progress.

Table 4-12 – Shell and Utilities Standards

Service	Type	Specification	Subclause	
Shell and Utilities	E	IEEE POSIX. 2	4.10.5.2	E
User Portability Extension (UPE)	E	IEEE POSIX. 2a	4.10.5.2	E
Control of interprocess communications, shared memory, and semaphores	E	IEEE POSIX. 4	4.10.5.2	E E
File transfer utilities, remote command execution, remote file printing, electronic mail, operating-system-based software development aids	G	X/Open XPG 3, OSF OSF /1, SVID , Berkeley BSD 4.x UNIX	4.10.5.3	E E E E

4.10.5.1 Current Standards

E

There are no currently completed or approved international or national standards for commands and utilities.

4.10.5.2 Emerging Standards

IEEE POSIX. 2 E

When completed, the IEEE POSIX. 2 standard will define a source code interface to command interpretation or shell services and common utility programs for application programs. These services and programs are complementary to those specified by POSIX .1 {2}.

The IEEE POSIX. 2a User Portability Extension will supplement POSIX. 2 by extending the specifications to promote the portability of users and programmers, in addition to applications, across conforming systems. Toward this end, the POSIX. 2a specifications expand the number and type of utilities specified, and enhance the features of a number of POSIX. 2-specified utilities, to provide a consistent interactive environment. The consistent interactive environment does not include emerging technologies such as graphical user interfaces, which are under development by different standards groups.

Parts of POSIX.2 go beyond the current service requirements and include a number of software development and debugging commands and utilities services. These are included in the POSIX.2 specification because of the traditional development orientation of UNIX systems. These software development and debugging services are not included in this clause because this clause includes more general and universal services, such as copying a file and reading a directory.

Although the POSIX.2 and POSIX.2a specifications are still in draft stages, they are relatively complete, and portions of the emerging standard are believed to be mature and stable.

When the commands, shell, and utilities specifications are completed and approved, the resulting IEEE POSIX.2 and POSIX.2a standards will be submitted to ISO/IEC JTC 1 for adoption as international standards. At that time, POSIX.2 and POSIX.2a will be combined into a single integrated international standard (ISO/IEC 9945-2).

IEEE P1003.15

When completed, the IEEE P1003.15 standard will provide batch queueing extensions to various POSIX base standards. These extensions define utilities, library routines, system administration interfaces, and an application-level protocol to address the following areas:

— Utilities for submission and management of requests

— System administration interfaces for the creation, management, and authorization of the network queueing and batch processing system

— language-independent programmatic (library) interfaces for application access to utilities and the queue and request database, and

— Application-level network protocols

4.10.5.3 Gaps in Available Standards

There are no formal interapplication standards that address the remote access and execution of a system's command interfaces. The Berkeley BSD UNIX de facto standard addresses all these service requirements, however.

4.10.5.3.1 Public Specifications

Public specifications that include the POSIX.2 and POSIX.2a, and go beyond these standards to also include the traditional UNIX-based command interfaces for electronic mail, remote command execution, file transfer, interprocess communications, shared memory, semaphores, and software development utilities are available from a number of organizations. These include:

— OSF's OSF/1 Application Environment Specifications (AES) E

— AT&T System V Interface Definition (SVID) E

— X/Open's XPG3 specifications, Volume 1 and part of Volume 3

4.10.6 POSIX OSE Cross-Category Services

4.10.6.1 Internationalization

The utilities described in the POSIX.2 specifications satisfy some requirements for standardized multilingual and multicultural support (e.g., localization requirements such as date formats and collation sequences, and support for international character sets). E

4.10.7 Related Standards

None. E

Section 5: POSIX OSE Cross-Category Services

Responsibility: Fritz Schulz

The POSIX reference model defines a set of conceptual system building blocks that collectively describes the Open System Environment. Each building block provides a specific set of interfaces for access to their associated facilities and services. There is another class of services and requirements, however, that may influence and/or impact the basic architectural building blocks; these are referred to as OSE Cross-Category Services.

An OSE Cross-Category Service is a set of tools and/or features that, when applied, may have a direct affect on the operation of one or more of the Open System Components, but it is not in and of itself a standalone OSE component. Examples of OSE Cross-Category Services include internationalization, security and privacy, administration, etc. Internationalization has a number of attributes that influence multiple OSE components; supporting multiple coded character sets, for example, will affect end-user interfaces, operational message input and output, screen display, data collating sequences in programming languages and database systems, etc.

This section will deal with the general characteristics of OSE Cross-Category Services as applied to the OSE architectural components and to the profiles and domains that characterize application environments. The specific impact/influence of an OSE Cross-Category Service will be described in the appropriate subclause of Section 4 that deals with individual OSE Components.

Initially, this section will address Internationalization, Security and Privacy, and System Administration; however, it is anticipated that other OSE Cross-Category Services will be identified as the concept is applied to the model.

This section describes issues that should be considered in writing profiles, and is organized so that subclauses for each OSE Cross-Category Service points to, and addresses issues adjacent to each of the service categories identified in Section 4.

These issues defined areas that need to be traded off to arrive at balanced solutions for a specific profile. It is expected that the specific trades would be made by the profiler, but that this clause could give guidance for trading and could also be used to accumulate lessons learned.

5.1 Internationalization

Responsibility: Ralph Barker

Editor's Note: Almost all instances of ''must'' in this clause have been changed to ''should'' without further diff marks.

5.1.1 Overview and Rationale

Historically, information systems intended for use within a particular national or cultural market have been designed specifically for the requirements of that market. If the vendor or developer was based in a country other than that of the target market, this was typically accomplished through substantial re-engineering the features of an existing system designed for some other country, and doing so at considerable cost. As the developer desired to market the system in additional countries, the process of re-engineering was repeated for each new national or cultural market. Application software developers were faced with the same problem. The very nature of this style of development produced little concern for portability across national or cultural boundaries, or interoperability between them. Users or organizations that needed to operate in multiple national or cultural markets typically did so with multiple, generally incompatible, information processing systems.

The interfaces provided by the POSIX Open System Environment (POSIX OSE) can be generalized, however, through the use of internationalization, to extend across national and cultural boundaries. Such a model provides the foundation for international portability of application software, increased user portability, and enhanced interoperability and data exchange capabilities. The task of internationalization is to ensure that the services provided by the POSIX OSE, and the interfaces between such services, are specified in such a way that they can be easily used all over the world. Additionally, as the user is likely to require services from any or all of the service categories of the POSIX OSE, internationalization impacts all areas of the POSIX OSE, and should be viewed as an OSE Cross-Category Service. Since the internationalization aspects of general OSE services and application program interface (API) services are similar for all of the POSIX OSE service categories, they are discussed here rather than repeating them in each of the services sections within this guide.

The ability of the service categories of the POSIX OSE to support multiple natural languages, and the underlying cultural conventions, is a two step process. These two steps are generally referred to as ''internationalization'' and ''localization.'' First, the interfaces between the service categories are generalized, so that they are not oriented to the requirements of any particular natural language or set of cultural conventions (internationalization). Then, facilities are provided by the POSIX OSE that allow the user to select the desired natural language and cultural conventions (localization). Tools are provided to facilitate this process.

Within this context, cultural conventions, while discussed more fully later in this clause, may be viewed as various aspects of how information is presented to the user. Different cultures, for example, use different formats for dates and numeric values and use different currency symbols. The interfaces provided by the POSIX OSE should allow the information to be presented to the user in the appropriate format as well as the appropriate natural language.

5.1.2 Scope

The POSIX OSE provides services that are necessary to support users, irrespective of their particular natural language or cultural conventions. While it is not expected that every implementation of the POSIX OSE would provide support for all possible natural languages and cultural conventions, the specification of the services and the interfaces within the POSIX OSE should not preclude such support. In addition to the service and interface requirements described here, it should be noted that internationalization is affected by a number of elements that are beyond the scope of this guide. Actual implementations of the internationalized POSIX OSE, for example, may need to consider the impact of multiple sets of governmental and regulatory agencies, international data communication standards and other elements which are presently not specified within the POSIX OSE, such as data portability between localized information processing systems.

Service requirements differ from country to country and even between users within one country. Many users, for example, may require the simultaneous support of multiple natural languages and cultural convention sets. Therefore, the basic internationalization requirement within the POSIX OSE is to provide a set of services and interfaces that allow the user to define, select, and change between different culturally related application operating environments supported by the particular implementation. Specifically:

— The POSIX OSE should provide the means of adjusting the output of specific functions and utilities to support different natural languages, cultural conventions and character sets as may be required by the supported natural languages.

— A user should have the capability to select an internationalized user environment that specifies a particular set of data presentation characteristics, including cultural conventions, character sets and native language.

— An implementation of the POSIX OSE should be able to concurrently support different applications functioning in different internationalized user environments, supplying different sets of natural languages, cultural conventions and character sets for different users.

— The capability of supporting different internationalized user environments, and the associated natural languages, cultural conventions and character sets, should not require any changes to the logic of existing application programs.

— The effect of the user selecting a new internationalized user environment, and its associated natural language, cultural conventions and character set, should be transparent to application programs.

— The model should be flexible, to support future extensions and requirements.

5.1.3 Reference Model

Internationalization is an OSE Cross-Category Service, spanning all OSE service categories. While various reference models have been used in published technical papers to depict internationalization issues, the internationalization services described in this clause conform to the POSIX OSE Reference Model.

5.1.4 Service Requirements

The POSIX OSE should provide services on different levels: general service requirements to be satisfied for any requesting program; API service requirements to be satisfied at the application program interface for a specific program; and a set of tools to support the localization of systems and applications. This subclause (5.1.4) will discuss these different service requirements in detail. In examining these service requirements, it is helpful to draw a distinction between those services which are required to support the portability of an application platform across cultural boundaries, and those services which are required to support the portability of an application across one or more sets of cultural conventions which may be supported on a single application platform.

5.1.4.1 General Service Requirements, Application Platform

Internationalization requirements are focused on support and handling of:

— Character sets and data representation

— Cultural conventions

— Natural language support

5.1.4.1.1 Character Sets and Data Representation

The character set for the English language can easily be satisfied by the standard ASCII character set (American Standard Code for Information Interchange). The ASCII code uses 7 bits to uniquely identify each of the 95 available characters. For European and American languages beside English, the number of local characters is much larger. The far-east requirements for thousands of pictograms add yet another dimension to the coding rules and techniques.

Different standards address the methods by which the local character repertoires can be coded for unique identification. While replacement of seldom-used characters in the 7-bit codings can support a single additional language besides English, 8-bit coding schemes are used to satisfy multiple languages concurrently by assigning an additional 96 graphic characters to the available repertoire. An example is ISO 8859-1 (the extended ASCII code), which can support all of western Europe, America, Australia, and other English speaking countries all over the world. For Eastern Europe, Greece, Russia, Arabia, and many other countries, other 8-bit codes are defined. Japan, China, Korea, and Taiwan have so many characters in their repertoire that 16 bits are needed to identify them clearly. Work is under way to develop a multi-octet character set with up to 32 bits per coded character; this method will allow concurrent use of all possible languages in the same application.

Because different coding schemes are used, it is important that the application platform have the potential capability of supporting all of them. It is also important that the application platform has the capability to represent (display, print) the data correctly. It is also important that an application be able to determine in which coded character set data items are stored on disk or tape. Otherwise, it is impossible for the application to interpret the data correctly. Currently the user must control the consistent use of the same coded character set within an application, but in the future the application platform should be able to provide identification methods for the coded character sets used for data storage, processing, communication, and presentation. It might also be advantageous for the application to be able to prohibit users from updating data stored in one coded character set with data in another coded character set since this would immediately corrupt data bases or flat files. Therefore it may be necessary in the future to provide a method of announcing the coded character set in which data are stored, processed, communicated, and presented.

The general service for support of character sets and data representation in an international environment are:

(1) Coded character set independence: the ability of the application platform to input, store, manipulate, retrieve, communicate, and present data independent from the coding scheme used. This includes 7-bit, 8-bit, 16-bit, and multi-octet coded character sets.

(2) Character set repository: the ability of the application platform to maintain and access a central character set repository. This repository contains all coded character sets used throughout the platform and specifies relevant information about them:

— Code format: the repository contains information, if characters are coded in 7 bits, 8 bits, 16 bits, or any other format.

— Data class definition: the definition that a character is considered numeric, alpha, etc., by the programming languages. This classification can vary for the same character from country to country.

— Collating rules: different character sets have different coding for characters. Thus, comparison of strings of such coded characters should follow rules defined for the specific character set. Culturally dependent additional collating rules are discussed in 5.1.4.1.2.

— Lower- to uppercase mapping: this defines the rules of mapping, if for a specific character no upper- or lowercase is available. Examples are the lower case umlauts which do not have uppercase representations in Switzerland; the uppercase forms are A, O, or U, respectively, followed by a lowercase ‘‘e’’.

— Escapement rules: some languages like Hebrew and Arabic are written from right to left; numbers within text in these languages are written from left to right. It is necessary to store these escapement rules with the character set.

— Presentation rules: the application platform should have the ability of providing fallback presentation rules for the presentation of coded characters that have no associated graphic shape.

(3) Character set identifier: the application platform should provide the ability to uniquely identify each coded character set to allow compatibility checks and translation or transliteration to and from other registered character sets. This ensures data integrity in the communication of data across computers and networks.

(4) Character set selection: the application platform should allow the end-user or the application to select the coded character set to be used; otherwise, the application should automatically select a default coded character set according to preset parameters. It should be possible to switch to other coded character sets and to invoke translation routines where required.

(5) Data announcement: the application platform could benefit from having the ability to recognize the coded character set of data entities (files, messages, etc.). One way of doing this is to store the character set identifier together with the data; standardization efforts are under way to formalize this process, with consideration being given to the level of granularity of such identification (e.g. file, word, character). The announcement enables the application to prohibit updates with data coded in other character sets, thus ensuring data integrity even in distributed systems.

(6) Data presentation: the application platform should be able to present data on different display or output devices, potentially according to rules in a repository, including escapement of characters and selection of different shapes. Preparing data for presentation may involve extensive translation and transliteration due to potential hardware limitations of the printers and displays used in a particular installation.

(7) Data communication: the application platform should be able to transmit and receive data from communication systems and to maintain the integrity of the information. In an internationalized environment, this capability might include data translation due to different coded character sets being used by different service categories of the application platform.

(8) Data input: the ability to enter data is not necessarily controlled by the application platform. The complexity of the input of Asian languages though might strongly support the idea of a standardized input mechanism interface. Depending on how other internationalization service requirements are met, it might also be beneficial for input data to carry some form of character set identification.

5.1.4.1.2 Cultural Conventions

Besides using different characters and different languages, countries throughout the world have also developed quite different cultural conventions. Even within one country we can find significantly different cultural environments. The prime example is Switzerland, where French, German, Italian, and Rhaeto Romanic are officially accepted languages. Combined with the language preferences are conventions about the formats of time, date, numeric values, and measuring systems. Currency symbols, paper formats, hyphenation, and collating are dependent on cultural conventions. End-user-oriented applications have to address these issues to provide a familiar local view, which helps to prevent operating errors.

The general service requirements for cultural conventions are:

(1) Cultural convention repository: The application platform should have the ability to store and access rules and conventions for cultural entities. These might be areas with a common language, geographic areas, or areas with common cultural or historic background. The repository should contain specifications and presentation rules for:

- Date and time formats: indicating the formats associated with the particular cultural entity. For example, while in the US the date is expressed in the format month/day/year, the European preferred format is year-month-day for data processing purposes and day-month-year in personal use. Japan counts the years according to the reign of the current emperor. Additionally, twenty-four-hour clocks, which are prevalent in Europe, are commonly used only in military circles in the US, while the terms ‘‘am’’and ‘‘pm’’,denoting morning and afternoon, are used by the general public. These are only a few examples for the cultural differences in this area. The application platform should be able to store the preferred forms for date and time for a specific cultural entity and make it available upon request in this format.

- Week and day numbering: in Europe, the week starts on Monday, in the US on Sunday. The application platform should be able to supply the requesting program with the needed information, potentially from a repository according to specified rules.

- Formats of numeric fields: handling of numeric fields in unfamiliar formats is one of the major reasons for human errors. The application platform should provide the service to format the values according to specifications in the repository. The characters that signify the decimal point (comma, period, etc.) should be defined, as well as the number of decimals, the grouping of digits before the decimal point and the presentation of negative values.

- Currency symbols and field length: the handling of currency symbols in the different cultural areas should be provided by the general internationalization services. The currency symbols might be more than one digit long and can appear before or after the currency field. The format of currency fields might differ from that of numeric fields; for example, in Portugal the $-sign is used as the decimal point. Information about these conventions should be stored in the repository and be used by the application platform for local formatting of currency fields. Not necessarily a service, but similarly important, is the understanding, that due to the value of different currencies, the field lengths should be considered carefully. Also some currencies do not have decimals (e.g., Italian Lira).

— Paper formats: internationally usable and portable applications should be able to print on different paper formats. While quart format is predominant in the US and the far east, the DIN standardized A-formats are used in Europe. Printer drivers should be able to adjust their output to local formats, defined in the cultural convention repository.

(2) Cultural repository selection: these repositories should be available to all applications. Users and applications should be able to select a repository from the application platform; a default value should be provided if no selection is made. An additional service allows dynamic switching to other repositories upon user or program requests.

(3) Collating rules: besides the generic binary and character-set-dependent sorting rules, the application platform should have the ability to sort data according to local rules, defined in the repository. An example for culture-dependent collating rules is the handling of umlauts; while they are sorted with the base characters in Austria, they are sorted at the end of the alphabet in Sweden. Adding complexity, they can be sorted differently within one country between normal business use, such as dictionaries, and in telephone books. Other idiosyncrasies are the sorting of one character as two (the German ‘‘sharp-s’’ sorts as ‘‘sz’’in Austria and ‘‘ss’’in Germany), or two characters as one (the Spanish ‘‘ch’’sorts as one character), or the position of accented characters in a string, and more. User-defined collating tables in the cultural convention repository allow culture or application-dependent sorting services.

5.1.4.1.3 Natural Language Support

The POSIX OSE should give users the ability to select a natural language for their dialogue with the system and applications. While it is unrealistic to expect all application platforms to support all possible natural languages, error messages, online documentation and help facilities, selection menus, and the relevant user interaction with these services should be prepared for translation into the supported user-selectable natural language. Additionally, the POSIX OSE should support differences between the natural language selected by the user for interaction with the application platform and that selected for use within a particular application. For word- and text-processing, the service includes hyphenation and spell checking with possible thesaurus support in different languages. The problem is complicated by the fact that data can contain text in different languages in the same document.

The service requirements for natural language support are:

(1) Multilingual capability: the application platform should be able to support more than one language simultaneously. For example, one process might be providing French language capabilities while another process operated in Japanese. The application platform should be able to let users select their preferred languages for communication with the application and allow them to switch dynamically to another language. The application platform also should have the capability to assign a default language, based on parameters for the application platform, the specific workstation, the user identification, or the application.

(2) Natural language message system: the application platform should have the capability to present (display, print, . . .) messages, menus, forms, and online documentation in the language, selected by the user. The application platform should be able to support multiple languages simultaneously for different users and it should allow the user to switch from one language to another. The following problems also should be handled correctly:

— The program code of the application should be able to be independent from any particular natural language, presenting messages in the natural language used within the internationalized user environment selected by the user.

— Variable message length: the application platform should support the presentation of messages of variable length, as translation into other languages changes the length of the message; English text is usually quite short compared to the same text in, e.g., German or Finnish. Ample room should be available in the display field to accommodate this variation.

— Inserted parameters and word order: the application platform should have the capability of inserting variable parameters into messages at the location appropriate for the user selected natural language.

(3) Support of local keyboards: the application platform should be able to correctly interpret the input from keyboards that have been modified locally to support the local character sets.

(4) Local language user interaction: the application should be able to accept solicited input from the user in the language selected by the user, without dependence within the application logic on a particular natural language or set of cultural conventions. For example, many applications use the first characters of prompts to make selections; this method is not acceptable in an internationalized system. The translation process changes the prompts and with them their first character; more than one prompt could have the same start-character and the program logic would not work. Multiple languages should be supported simultaneously.

5.1.4.2 API Service Requirements

All the general services defined in 5.1.4.1 should be accessible from the applications through requests to the application program interface. The API service requirements can be structured in the same way as the general requirements, which they call for.

5.1.4.2.1 Cultural Conventions

- — Cultural convention invocation: the application platform should allow the application to invoke a specific cultural convention from the repository. It should automatically invoke the default convention set, if no selection is made by the application.
- — Cultural convention change: when requested by the application or the user, the application platform should change the used cultural convention dynamically.
- — Provide local values: upon request from the application, the application platform should return local formats for time, date, calendar, numeric fields, currency fields and symbols.
- — Local sort and comparison: when requested by the application, the application platform should compare and sort data according to the local collating rules defined in the cultural convention repository.

5.1.4.2.2 Natural Language Support

- — Language selection: the application platform should present messages, menus, forms, online documentation, and user interaction in the natural language selected by the user or automatically by the system based on preset parameters for the application, the session, the user, or the system.
- — Change of language: upon request from the user, the application platform should be able to dynamically change, prior to the invocation of a particular user application, the language used for messages, menus, forms, online documentation, and user interactions.

5.1.4.3 Localization Tools Requirements

Internationalization of application platforms and applications is the basis for their localization in the different countries. It is important for the user that this localization can be performed in a well prepared, organized way without the need to know the internal structure of the application platform or the application. The following requirements for localization tools are key to successful localization of application platforms and applications:

- — Character set repository tools: tools should be provided to set up and maintain character set repositories. They also should allow the addition of new character sets to the repository.

- Cultural convention repository tools: tools should be provided to set up and maintain the cultural convention repositories. Addition of new cultural environments should be possible. User-definable collation tables are essential parts of these repositories; tools to define and maintain them should be offered.
- Translation support tools: facilities for the set-up and maintenance of local language message files, menus, forms, online documentation, and user interaction tables should be provided. The addition of new supported languages should be allowed by such tools. Additionally, any such translation tools should allow revision control, so that only new or changed text would require translation for new software releases.

5.1.5 Standards, Specifications, and Gaps

There are not many standards available that deal with internationalization. The majority of current standards describe character sets, both for control characters and for graphic characters in different coding schemes (7-bit, 8-bit, etc.). A few standards address the formats of time and date, and some standards touch peripherally on the subject of data announcement.

An example of how cultural conventions and languages are currently supported is the *locale* () function. It allows the application developer to select portions or all of predefined support features for national languages and local cultural conventions. The portions, called categories, correspond to the areas of functionality; presently supported are character classification, collation sequence, date/time format, monetary format, and numeric format. Other categories, such as message handling, are likely to be implemented, too. Other systems have started to implement similar philosophies of general services to support local cultural conventions.

5.1.5.1 Current Standards

5.1.5.1.1 International Standards

- ISO 646: 1983, *ISO 7-Bit Coded Character Set for Information Interchange*

  Defines the binary representation of 128 control, (Latin) alphabet, digit, and symbol characters. Describes in general the use of the control characters. Describes option of national replacement characters.

- ISO 2014: 1976, *Writing of Calendar Dates in All-numeric Form*

  This international standard specifies the writing of dates of the Gregorian calendar in all-numeric form, signified by the elements year, month, and day.

- ISO 2022: 1986, *ISO 7-Bit and 8-Bit Coded Character Sets—Code Extension Techniques*

Table 5-1 – Internationalization Standards E

Service	Type	Specification	Subclause	
Character set/data representation	S	ISO 646, ISO 2022, ISO 4031, ISO 4217, ISO 4873, ISO 6093, ISO 6429, ISO 6936, ISO 6937-1, ISO 6937-2, ISO 7350, ISO 8601, ISO 8859-*n* (1-9), CCITT T.61, GB 2312, JIS X 0208, KS C 5601	5.1.5.1	E
Character set/data representation	E	ISO DIS 10367, ISO DIS 10646	5.1.5.2	E
Cultural convention	S	ISO 2014, ISO 3307	5.1.5.1	E
Natural language support	E	ISO/IEC 9995-x, CSA-Z243.200-88	5.1.5.2	E

Defines techniques for expanding the number of characters represented by the base character set.

— ISO 3307: 1975, *Representation of Time of the Day*

This international standard is designed to establish uniform time representation based upon the 24-hour timekeeping system. It provides a means for representing local time of the day and Universal Time in digital form for the purpose of interchanging information among data systems.

— ISO 4031: 1987, *Representation of Local Time Differentials*

This international standard specifies a standard means for representing local time differentials to facilitate interchange of data among data systems.

— ISO 4217: 1987, *Codes for the Representation of Currencies and Funds*

Specifies the representation of currencies and currency symbols

— ISO 4873: 1986, *ISO 8-Bit Code for Information Interchange—Structure and Rules for Implementation*

Outlines the structure of the ISO 8-bit code and rules for implementation.

— ISO 6093: 1985, *Presentation of Numerical Values in Character Strings for Information Interchange*

Specifies three presentations of numerical values, which are represented in character strings in a form readable by machine, for use in interchange between data processing systems. Also provides guidance for developers of programming languages standards and Implementor's of programming products. These representations are recognizable by humans, and thus may be useful in communication between humans.

— ISO 6429: 1988, *ISO 7-Bit and 8-Bit Coded Character Sets—Control Functions for Coded Character Sets*

Defines control functions and their coded representations for use in a 7-bit code, an extended 7-bit code, an 8-bit code, or an extended 8-bit code. Specifies a C0 set, a C1 set, control functions derived there from, and a number of independent control functions.

— ISO 6936: 1988, *Conversion between the Two Coded Character Sets of ISO 646 and ISO 6937-2 and the CCITT International Telegraph Alphabet No. (ITA) 2*

Specifies the rules for conversion between ITA 2 representation of 58 characters and the ISO 646 representation of 128 characters.

— ISO 6937-1: 1983, *Coded Character Sets for Text Communication—Part 1: General Introduction*

Defines terms and concepts used in describing and using code representations of character sets.

— ISO 6937-2: 1983, *Coded Character Sets for Text Communication—Part 2: Latin Alphabetic and Non-alphabetic Graphic Characters*

Defines a repertoire of Latin alphabetic and non-alphabetic characters. Specifies binary representation of the characters. Specifies rules for the definition and use of character sets that are subsets of the repertoire.

— ISO 7350: 1984, *Registration of Graphic Character Subrepertoires*

Specifies the procedures for preparing, registering, publishing, and maintaining the register of graphic character sets that are composed from the character repertoire of ISO 6937 and the procedures for assigning identifiers to the sets.

— ISO 8601: 1988, *Representation of Dates and Times*

Specifies the representation of dates A.D. in the Gregorian calendar and times and representation of periods of times. Applicable whenever dates and times are included in information interchange.

— ISO 8859-x: 1987, *8-Bit Single-Byte Coded Graphic Character Sets*

Specifies a set of up to 191 graphic characters by means of a single 8- bit byte. The versions (‘‘-x’’) indicate different coded character sets:

-1 Latin Alphabet No. 1

-2 Latin Alphabet No. 2

-3 Latin Alphabet No. 3

-4 Latin Alphabet No. 4

-5 Latin/Cyrillic Alphabet

-6 Latin/Arabic Alphabet

-7 Latin Greek Alphabet

-8 Latin/Hebrew Alphabet

-9 Latin Alphabet No. 5 E

— CCITT T.61, 1985: *Character Repertoire and Coded Character Sets for the International Teletex Service*

Describes detailed definitions of the repertoires of graphic characters and control functions to be used in the international Teletex service. The means by which supplementary character repertoires are defined are also described.

5.1.5.1.2 Regional Standards

Presently, no regional internationalization standards which relate to the scope of this guide have been adopted.

5.1.5.1.3 National Standards

Many of the international ISO standards have ''twins'' in the national standards bodies; i.e., the same text is given a local standard identification. Also, national standards bodies have often developed standards for local representation of time, date, and currency. The implementation of these standards into an internationalized system is a prime example of localization.

Here are some standards that have no international equivalent:

— GB 2312: 1980, Chinese national character set standard

— JIS X 0208: 1983, Japanese national character set standard

— KS C 5601: 1987, Korean national character set standard

5.1.5.2 Emerging Standards

5.1.5.2.1 International Standards

The rapid development of business opportunities in the Pan-European and the Asian market has spawned a wealth of activities to develop standards for the support of internationalization in the field of information technology. These emerging standards deal with character sets, language neutral user interfaces, and communication.

— ISO DIS 10646: *Multiple Octet Coded Character Set*

This standard will permit the presentation of all of the world's scripts in computer based systems, and their unambiguous interchange between one system or person and another. It is applicable to the representation, processing, storage and presentation of the written form of the languages of the world.

— ISO/IEC DIS 10367: *Repertoire of Standardized Coded Graphic Character Sets for Use in 8-Bit Codes*

This standard specifies a unique graphic character set for use as G0 set and a series of coded graphic character sets of up to 96 characters for use as the G1, G2, and G3 sets in versions of ISO 4873. All sets specified in this standard are shown as elements of an 8-bit code.

— ISO/IEC CD 9995-x: *Information Technology—Keyboard Layouts for Text and Office Systems*

This family of standards defines the layout of keyboards so that they can be used for input of multilingual information.

5.1.5.2.2 Regional Standards

The European Community is in the process to define European standards, called EN (Europaeische Norm). No internationalization standards have yet been adopted.

5.1.5.2.3 National Standards

National standards under development which relate to internationalization include:

— CSA-Z243.200-88: *Canadian National Keyboard Standard for the English and French Languages in Text and Office Systems*

5.1.5.3 Gaps in Available Standards

5.1.5.3.1 Public Specifications

The PC character set was defined at a time, when the international standards for single-byte, 8-bit character sets were not available yet. Therefore, the PC character set was accepted and still is a de facto standard in the PC world. The concept of different code pages has been implemented in MS-DOS and WINDOWS-3 is using ISO 8859-1 internally for compatibility reasons with other systems. Some companies have gone similar routes and developed their own, multilingual character sets for specific applications, the general trend is clearly towards ISO standards wherever they exist.

A consortium of software and hardware companies is developing ‘‘Unicode,’’ a 16-bit character set standard for broad international use. E

5.1.5.3.2 Unsatisfied Service Requirements E

While the character set arena is heavily populated, very little work is done in other areas of internationalization of products. Standards should be developed for:

— Cultural conventions repository

— Application program interface services for cultural conventions

— Application program interface services for character set handling

— Multilingual collating rules

— Input methods interface for Asian languages

— Standards for message delivery systems

— Data announcement standards

Additionally, no standards currently exist that support the following character set and data representation functionality:

(1) Character set invocation: the application platform should allow the application to invoke a specific character set from the character set repository. It should automatically invoke the default character set, if no selection is made by the application.

(2) Character set changes: When requested by the application, the character set should be changed dynamically.

(3) Character set identifier: the application program should be able to write the character set identifier to data and should be able to retrieve the identifier for requested data.

(4) Character set identifier comparison: the application platform should, upon request from the application or automatically, compare the character set identifiers of interacting data in the application (input, processing, data storage, communication, and output).

(5) Character set translation: the application platform should provide translation of character sets, when requested by the application or automatically, when detecting a mismatch in the comparison process.

5.1.6 OSE Cross-Category Services

Not applicable.

5.1.7 Related Standards

The nature of internationalization as being a cross-component facility is that it affects just about every element in the information processing world. Thus, almost all standards in this environment are related to the subject. Here we will point out a few major families of standards, strongly related to internationalization.

— ISO DIS 8613: Office Document Architecture and Interchange Format (ODA)

This family of standards, ODA/ODIF, consist of:

1.2 Introduction and General Principles

2.2 Document Structures

3 Document Processing Reference Model

4.2 Document Profile

5.2 Office Document Interchange Format

6.2 Character Content Architectures

7 Raster Graphics Content Architectures

8 Geometric Graphics Content Architectures

— ISO 8824: 1987, *Specification of Abstract Syntax Notation One ASN. 1*

Specifies a notation for the definition of abstract syntaxes, enabling Application Layer standards to define the types of information they need to transfer using the Presentation service. It also specifies a notation for the specification of values of a defined type.

— ISO 8825: 1987, *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN. 1)*

Defines a set of encoding rules that can be applied to values of types defined using the notation specified in ASN. 1. Application of these encoding rules produce a transfer syntax for such values. It is implicit in the specification of these encoding rules that they are also be used for decoding.

— All programming language standards, since programming languages have to support internationalization, and have to work correctly in localized environments. Their generated code itself has to work ‘‘localized.’’

E

5.2 System Security Services

Responsibility: Michelle Aden

5.2.1 Overview and Rationale

Information is the key to successful use of a system. For example, if used effectively and efficiently, information may be used to underpin enhanced service and to aid the derivation of strategic plans. Much of this information, for example, personal customer details and business financial plans, will be of a sensitive nature.

Although authorized users may be able to take advantage of the POSIX Open System Environment (OSE) to increase productivity and efficiency, unauthorized individuals may also be able to take advantage of the OSE to steal, manipulate or to deny others access to information held within the system, or to deny involvement in some transaction performed via the system.

Security services must therefore be provided within the system if it is to prevent these unauthorized activities. To achieve an optimum degree of confidence in the correctness and effectiveness of a system's security services, a system specific security policy must be derived and appropriate security functionality designed into the system at the beginning of its life cycle.

A relatively high degree of protection for ordinary computer systems can be achieved if system administrators correctly configure and maintain the system according to recommended security guidelines and practice, such as those described within the *X/Open Security Guide* . However, additional security facilities must be supported within the system to achieve protection against the small percentage of attackers who are noncasual, and who are determined to breach the security of the system. It is the intent of the security extensions to the base POSIX interface standard to support these additional security facilities.

The four basic security objectives of a system are to maintain:

— Confidentiality. The system must prevent unauthorized viewing of data.

— Integrity. The system must prevent unauthorized alteration or deletion of data.

— Availability. The system must ensure that authorized users are not prevented from accessing and processing data.

— Accountability. The system must ensure that users are made accountable for their actions, for example to ensure that users are correctly billed for system usage. See also 5.3.4.11. E

Different user groups may place different emphases upon these four basic security objectives. For example, the military security sector may place more importance upon confidentiality than accountability while, correspondingly, the commercial sector may place more importance upon accountability than confidentiality.

5.2.2 Scope

One of the goals of system security is to provide defense in depth, such that if one layer of security is breached then further layers of security will limit and/or prevent unauthorized activities within the system.

To achieve a high degree of confidence in the correctness and effectiveness of the security of a system that will be processing sensitive information, security must be designed into the system at the beginning of its life cycle.

A System Security Policy (SSP) defines what it means for a specific system to be ''secure'' and, as such, forms the basic security input into the system lifecycle. Specification of an SSP is therefore axiomatic to the design of a secure system.

Although the SSP defines what security measures will be provided within the system, it is the system design documentation that defines how these security measures will actually be implemented.

One aspect of an SSP may be that it mandates conformance with the POSIX security extensions.

Security interface specifications are intended to assist in the construction of a secure system. They do not, in isolation, provide any protection against threats to a system.

5.2.3 Reference Model

The reference model for security is the same as the model shown in Figure 3-3. Security has an impact on all of the APIs and EEIs in the model.

5.2.4 Service Requirements

Through an analysis of the potential threats and requirements of the system, the system security objectives and hence the necessary System Security Policy (SSP) rules may be derived. This analysis must also take into account appropriate corporate, legal, and standardization requirements.

System confidentiality, integrity, availability, and accountability may be supported by the following security objectives:

Technical Security Objectives

— Identification and Authentication. A system entity, such as a user or system element, must prove that its claimed identity is legitimate, such that another system entity may place confidence in that claimed identity.

— Access Control. Access to system resources will be restricted to authorized entities only. Residual data contained within an object will be securely erased before it may be reused by a system entity.

— Accountability and Audit. System users must be made accountable for their actions. Audit trails of these actions will then be maintained and utilized such that unauthorized system activity will be detected.

- Accuracy. The system must ensure that the correctness and consistency of security-relevant information is maintained.
- Availability. System resources will be provided to users in a consistent and reliable manner.
- Data Exchange. Data transmitted between system users and/or elements will be protected from unauthorized interference or viewing. Originators and recipients of data will be authenticated and will be able to mutually prove their respective participation in the transaction.

Nontechnical Security Objectives

- Assurance. The security of the system must be specified, designed, implemented, tested, and maintained in such a way that confidence can be placed in the correct and effective operation of the system. Also, procedures must be specified to ensure continued confidence in the security of the system in the event that the system is modified in some manner.
- Security Roles and Responsibilities. Security activities must be partitioned and allocated to identifiable security administrators who will then be responsible for ensuring that their allocated task is satisfactorily performed.
- Secure Operating Procedures. Procedures must be written that will guide system administrators and users as to the correct procedure to follow in the event of some security-relevant occurrence.

5.2.4.1 Application Programming Interface Services

The POSIX security interfaces will support Audit, Privilege, Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Information Labels (ILs).

The audit services include:

- Ability to record the user identification for actions within an audit trail
- Ability to process the audit trail
- Ability to use the audit trail to generate alarms

The privilege control services include:

- Ability to grant users only the minimal security required to perform a task

This will minimize the impact of a subverted security administrator or unauthorized usage of a security administrator role.

The discretionary access controls (DAC) provide the following services:

- Ability to control fine-grained user access to objects
- Ability to provide extended user access bits beyond the traditional user-group-other
- Ability to support access control lists (ACL)

The mandatory access controls (MAC) and information labels (IL) support policies for labeling:

- Ability to associate a MAC label with an object
- Ability to label information (e.g., physical document handling restrictions)

5.2.4.2 External Environment Interface Services

Note to reviewers: This subclause will be provided in a later draft. Mock ballot reviewers are welcome to submit comments on the types of services required at the EEI.

5.2.5 Standards, Specifications, and Gaps

Table 5-2 lists the current, emerging, and gaps in security standards.

Table 5-2 – Security Standards

Service	Type	Specification	Subclause
System Security	E	IEEE P1003.6 API	5.2.5.2
Access Control	E	ISO/IEC 8613	5.2.5.2
Directory Authorization	S	CCITT X.509	5.2.5.1
Security	G	ECMA CMA 138	5.2.5.3
Trusted Systems	G	DOD 5200.28- STD	5.2.5.3

5.2.5.1 Current Standards

ISO 7498-2, *Information Processing Systems—Open Systems Interconnection Reference Model, Security Architecture.*

ISO/IEC 8613, *Information Technology—Text and Office Systems—Office Document Architecture (ODA) and Interchange Format.*

CCITT X.509, *Message Handling System, ISO/CCITT X.400 Directory Authentication Framework.*

ECMA CMA 138, *Security In Open Systems—Data Elements and Service Definitions.*

5.2.5.2 Emerging Standards

Information Retrieval, Transfer and Management For OSI—Draft Access Control Framework, ISO / IEC SC 21/ WG 1.

Draft Addendum to ISO 8613 On Security

The P1003.6 scope is limited to security extensions for those interfaces defined within the base POSIX interface specification (POSIX .1 {2}). Issues not addressed within the P1003.6 scope include noninterface-specific architectural assurance issues and communications security.

5.2.5.3 Gaps in Available Standards

The Information Technology Security Evaluation Criteria, Version 1.2, 28 June 1991.

US DoD, DOD 5200.28- STD, *Trusted Computer System Evaluation Criteria.*

Trusted Network Interpretation

Trusted Database Interpretation

Computer Security Subsystem Interpretation

5.3 Information System Management

Responsibility: Don Folland, Neil Croft

5.3.1 Overview and Rationale

Information System Management issues are considered in this clause. The subject is concerned with the effective management and control of the complete set of resources that comprise an information system. The tools in support of the services required by system managers need to reflect the portability and interworking attributes of open systems and fit the Open System Environment Reference Model (Figure 3-3). It is necessary to consider a variety of system management support scenarios (central management, dispersed management, or hybrid), addressing both distributed systems and standalone systems. The issues apply to application software or software components of the application platform. It is necessary to support automated management and operation of the IT infrastructure and address a wide variety of licensing scenarios.

5.3.2 Scope

This category includes services and policies that address the administration of the overall information system required by any organization, including:

- Information Management
- Processor Management (e.g., Add new user)
- Network Management
- Configuration Management
- Security Management (e.g., Authentication, Key Management)
- Accounting Management
- Performance Management

Administration services accessible from the API may have Programming Language or Language Binding service specifications associated with them.

These services are defined to provide system and network administrator portability.

5.3.3 Reference Model

The Reference Model for system management is the same as the model shown in Figure 3-3. System management impacts all of the APIs and EEIs in the POSIX Open System Environment Reference Model.

5.3.4 Service Requirements

The following services should be provided: E

5.3.4.1 Processor Configuration Management

Configuration management consists of four basic functions: identification, control, status accounting, and verification.

Identification involves specifying and identifying all components of an IT infrastructure.

Control implies the ability to agree and ''freeze'' configuration items (CIs) and then to make changes only with agreement of the appropriate named authorities. Control is concerned with ensuring that none of the CIs shown is altered or replaced and that no CIs are added without appropriate authorization.

Status accounting involves the recording and reporting of all current and historical data concerned with each CI. Status accounting maintains records of the current, previous and planned states and attributes of the CIs and tracks these states and attributes: for example, as the status of a CI changes from ''development'' through to ''test,'' ''scheduled to go live,'' ''live,''and through to ''archived.''

Verification consists of a series of reviews and audits to ensure that there is conformity between all CIs and the authorized state of CIs as recorded in the configuration management database (CMDB). It is concerned with checking that the physical CIs actually match the authorized system as described in the CMDB.

5.3.4.2 Network Configuration Management

To ensure the viability of network services the configuration of systems and services must be controlled and managed. Effective configuration management will produce a minimum risk environment.

Configuration management procedures must ensure that details are provided for network equipment and systems covering:

- — Configuration activities—how to configure the network equipment
- — Security controls
- — Access controls
- — Configuration history log
- — Configuration authority
- — Build details
- — Fall-back and test records
- — Management reporting requirements.

5.3.4.3 Distributed System Configuration Management

The services here consist of the following:

- Authentication services for a distributed system environment
- Distributed Naming Service Configuration
- Distributed Time Service Configuration
- X Window system configuration
- Window/Session Manager configuration

5.3.4.4 Software Installation and Distribution

The main types of software to be installed and distributed are application programs developed in-house, bought-in applications, and utility software and personal computer software packages. All software needs to be managed effectively from development or purchase through to the live environment. Unless the distribution and implementation process can be controlled automatically, or from the center using software tools, procedures must be in place to ensure that distributed software arrives when expected and is checked for authenticity in whatever way is practical, and that the software is brought into use when required. The main procedures involved in software distribution and installation are:

- System management staff at the center to inform remote staff when to expect distribution software to arrive.
- Recipients to report to system management staff when the distributed software has arrived successfully.
- System management staff to check that all software is received as expected at locations.
- System management staff to issue clear instructions about when the software is to be implemented.
- Location staff to report to system management at the center when the software has been implemented. The release record on the Configuration Management Database will state which installations are to receive the release. This database must be updated to reflect the receipt and implementation of the release at each site.

5.3.4.5 License Services

The terms and conditions relating to the supply of software may place legal restrictions on the organization (e.g., no unauthorized copies to be made). It is particularly important therefore that the Configuration Management Database is updated with details of who holds copies of software items. This assists the organization in discharging its legal obligations and assists auditors in checking for the existence of unauthorized copies.

All authorized copies of licensed or purchased software that are made by system management staff should be allocated a unique copy number and recorded in the Configuration Management Database together with where they are located and who is responsible for them. Procedural restrictions should be introduced to prohibit the unauthorized copying of software, and regular software audits should include a check for any unauthorized copies.

5.3.4.6 Print Output and Distribution Services

Output and distribution packages control output production and distribution from the moment the output is planned to the time the user receives the print. The working criteria need to be set up first; e.g., define who receives the report and how much of the report the user gets.

The main functions are:

- The report can be limited to parts wanted by the user.
- Multiple copies of the entire report, or of selected sections can be produced.
- Reports are grouped by recipient within delivery location.
- Reports for each job are spooled as a group when the job is complete.
- The number of whole reports and individual pages received by each user are recorded.
- Report production can be monitored and managed efficiently.

Output and Distribution packages should include the following:

- Printing and distribution of whole and part reports
- Status (queued, printing etc) of the report tracked
- Online viewing of reports
- Ability to archive report files
- Ability to support a wide range of printers
- Costing and charging functionality
- Security facilities

By using an output distribution package, the delivery of reports to the correct person at the correct location can be ensured. Paper, time, and IT resource are saved as the users receive only the parts of reports that they need, and can also view the reports online. The number of pages printed can be controlled. Reports can be tracked from the time they are created to the time they are delivered to the user, allowing good security monitoring.

5.3.4.7 Office Media Management and Backup/Restore

The main services of magnetic tape and data cartridge management systems are: E

- Provide automated support for tape housekeeping and maintenance including:
  - Allocating tapes and releasing them for reuse helping
  - To ensure even patterns of use where appropriate
  - Constructing and triggering cleaning schedules
  - Maintaining the security of data
- Help automate archiving (vault management) for offsite storage
- Help identify growth requirements

Vault management is concerned with controlling the movement of tape cycles from one storage location to another. As a tape cycle is used, the tape management system automatically logs a different vault identifier against each tape.

A backup strategy is required to control the frequency of backups and the way in which they are created; e.g., whole volumes to cartridge or individual files to tape.

The backups and restores of system and application software should be separate from the backups and restores of data. Software and library backups should be explicitly scheduled and the complete software item or library backed up. The schedule for backing up files must be fully documented, properly maintained and adequately safeguarded as the contents of the schedule are required for disaster recovery purposes.

5.3.4.8 Online Disk Management

The operation of disk management systems requires that they take account of a range of factors such as retention period, recovery, space fragmentation, disk overflow, file and record activity levels, and channel use. Some systems merely report against values or thresholds set, but increasingly they invoke corrective action. Typically, the corrective action is file and disk reorganization or file and data archiving.

If a disk management system is used, the constant monitoring and actioning of requests for disk space can be minimized. Disk space may be collectively pooled and unused space constantly reclaimed.

5.3.4.9 Job Scheduling

Scheduling involves the continuous organization of jobs and processes into the most efficient sequence, maximizing throughput and utilization to meet the targets set in service level agreements (SLA). Jobs are scheduled to ensure:

- SLAs and user requirements are met; e.g., certain jobs need to be run by a certain time

— Available capacity is used effectively; e.g., the workload run at any given time does not exceed the practical capacity.

The minimum services of a scheduler should include:

— A high upper limit for the number of relationships allowed between jobs

— The ability to schedule by calendar and criteria

— Workload balancing support

— Levels of security

— Ability to restart jobs

— Operator override capability

— Capability to model future workloads.

5.3.4.10 User Administration

The services here consist of the ability to:

— Create a new user or group of users

— Delete a user or group of users

— Allocate system resources to a user or a group of users

5.3.4.11 Accounting

An effective cost management system should contribute to the development of a sound investment strategy that recognizes and evaluates the options and flexibility available from modern technology. The services here should provide the ability to:

— Establish targets for performance

— Measure performance against targets

— Measure and prioritize resource usage

— Monitor assets and maintain records for control purposes

— Apportion costs of IT services to users

— Report costs to management and users

5.3.4.12 Performance Management

The services here should provide the ability to:

— Monitor hardware, software, and network performance

— Monitor workload and throughput

— Set and adjust system parameters to tune performance

— Monitor terminal response time E

5.3.4.13 Capacity Management

An effective and efficient capacity management function contains at least the following elements:

- — Performance management to monitor and optimize the use of current systems.
- — A capacity management database that contains current and historic data of technical and business related interest. This database forms the basis for the provision of both tactical and strategic reports on performance and capacity.
- — Workload management to identify and understand the applications that make use of the system. The understanding of workloads has both a technical and business related nature. This involves application sizing to accurately predict the performance and required capacity of new applications.
- — Capacity planning to accurately plan the required hardware resource and associated cost for the future and to predict the effect on performance and capacity of both tactical and strategic plans.

5.3.4.14 Fault Management E

These services allow the system to react to the loss or incorrect operation of system components at various levels (hardware, logical, services, etc.). The classical model of fault tolerance has a three-step approach. The three steps are fault detection, fault isolation, and fault recovery. Typically implementations divide these steps into multiple steps or integrate them into one or two steps. Additionally, fault diagnosis services support the other steps in the treatment of a fault.

Various fault tolerance strategies, such as checkpointing and voting, are implemented as a collection of services comprising one or more of the steps in the fault tolerance classical model. For example, services involved in implementing a three-node voting scheme will include a vote comparator service (fault detection), vote analyzer service (fault isolation/fault diagnosis), a service to pass the majority ‘‘answer’’ through (fault recovery) as well as a service to disable the faulty resource and reconfigure the voters (fault recovery/reconfiguration).

Fault Detection

Fault detection services are concerned with determining when a fault has occurred in the system. Fault detection services are both passive and active. Active services are those that attempt to determine the status of various system components by testing those components. Passive services, on the other hand, try to ascertain system components by passively gathering information and watching the behavior of the system.

Fault Isolation

Fault isolation services attempt to determine the component at fault and segregate the faulty component from the rest of the system. Services may be shared between the fault detection and isolation service library in that they perform both functions.

Fault Recovery

Fault recovery services attempt to bring the system into a consistent state. These services may be very interrelated to the scheduling services, network services, and data base services, depending on the recovery scheme used.

Redundancy of resources is many times needed to support fault recovery. Resources may include data, process, processor, disk drive, etc.

As parts of the system fail, it may no longer be possible to satisfy all the requirements of the application. Services to support graceful degradation may be used to ensure that critical activities do not fail.

Fault Diagnosis

These services deal with the system's ability to analyze the attributes of a system fault and determine its cause. These services tend to be very interrelated with fault detection and fault isolation services.

Fault Avoidance

These services involve the avoidance of faults before a failure in the system component occurs. If a system can detect that the operation of a component is approaching the edge of its operational range, a standby or backup component could be phased in to replace it. Another form of fault avoidance is logging of shocks, temperature extremes, etc., so that it can be predicted that a component will not meet its original expected service life.

Software Safety

These services involve the system's ability to keep application software from causing harm to the system's software, hardware, or user. For instance, a process may attempt to write into another process's memory space without permission.

A good example of a reliability method that may provide software safety is a bounds checker. The checker compares an answer supplied against the bounds. If it is not within the bounds, the bounds checker will not allow the answer to propagate, possibly causing damage to the system's integrity. Additionally, it may send a fault message (or security violation information, depending on the type of answers expected) to the proper service.

To enhance software safety, other services and processes should be only given the resources necessary to complete their job.

Status of System Components

These services involve the obtrusive and nonobtrusive diagnosis of the state of system components. For further explanation of these services, see Fault Detection and Fault Diagnosis services. These services may additionally need to record and/or display information concerning performance, configuration, and general system information.

Reconfiguration

These services allow the system to reconfigure its view of the world. This services allow the system to substitute different resources to perform system functions such as substituting a new physical I/O channel to support a logical channel. These services are part of the API but their use may be restricted to specially authorized programs such as those used by the target system operator.

Maintainability

Maintainability services provide support for the maintenance of a system. A major component of that support is the collection and logging of information about the operation of the system. Typical information to be logged is:

— Software and hardware errors during operation

— Processes that failed or almost failed to meet scheduled deadlines

— Performance metrics for system tuning

— Times when the system operated in extreme environmental conditions

— Errors reported during startup self-testing

— Attempts to violate rules of the system's security policy.

5.3.4.15 Security Management

— Configuration of appropriate ACLs for System, User Interface, Storage, Network, and application software services.

5.3.5 Standards, Specifications, and Gaps

There are a number of international and national initiatives to develop standards for system management.

Note to reviewers: This subclause will be expanded in a later draft.

5.3.6 OSE Cross-Category Services

— Security for remote print jobs

5.3.7 Related Standards

None. E

Section 6: Profiles

Responsibility: Fritz Schulz

This section targets those who want to know more about what profiles are and those who are in the process of developing their own profiles. The latter group consists of those developing formal ''Standardized Profiles'' and those developing less formal profiles for their industry group (e.g., a banking trade association) or their own company or enterprise for procurement or strategic planning purposes.

Those not involved in the development of profiles should read 6.2. Parts of 6.3 also may be useful, especially the earlier subclauses that give definitions of terms and explain concepts more precisely.

Developers of profiles that are not formal POSIX Standardized Profiles (POSIX SPs) should read all of Section 6.

Developers of profiles that are formal POSIX SPs should read all of Section 6 and Annex A.

6.1 Scope

The information presented here about profiles is limited in scope to assist those needing to understand profile concepts as they apply to the POSIX Open System Environment. Covered are profiles constructed from standards (and profiles) listed within this guide (that, by design, are consistent with POSIX.1).

The goal is to create a common approach and documentation scope and style for POSIX-oriented profiles. Annex A goes further by giving specific guidance to developers of formal POSIX SPs.

6.2 Profile Concepts

Responsibility: Bob Gambrel

Introduction

This guide is designed to assist in the selection of standards in the procurement process or as a target application environment. Profiles also assist in the selection of standards. A profile is a suite of base standards with specified options. Profiles can be created by software developers to describe the environment they target or by buyers to identify their purchasing objectives.

Basic Terminology

There are two general classes of standards documents:

- Base standards
- Profiles, including application environment profiles (AEP), standardized profiles, and POSIX standardized profiles

See 2.2.2 for format definitions of these terms. As used in this guide, base standards specify functionality, syntax, protocols, data formats, etc., in detail, while profiles do not. Instead, profiles (sometimes called ''functional standards'') identify which base standards are applicable. Since base standards often consist of a base or mandatory part and a number of selectable optional parts and values, profiles may also (or may not) choose, for each base standard, specific options or values. A profile may also identify other profiles, allowing the construction of ''larger'' profiles based on both base standards and other ''smaller'' profiles.

NOTE: In the context of internationalization, the term ''national profile'' is frequently used and will be found, for example, in POSIX.1 {2} and POSIX.2. Its meaning is consistent with the definitions in 2.2.2, but in many cases such profiles reflect national cultural conventions. For example, Denmark and Japan both have specified a national character profile.

6.2.1 Relationships Between This Guide and Profiles

Key to the understanding of profiles is a discussion of the relationships that exist among profiles, this guide, and the base standards.

There exist many thousands of base standards, each addressing a particular, usually narrowly scoped, area of application portability or interoperability. Many of the base standards, developed over the years, are simultaneously narrow in scope (for example, a C binding of SQL), but broadly applicable (for example, applicable to operating systems that comply with POSIX specifications and those that do not.)

The base standards listed in 1.2 form the basis of the POSIX Open System Environment. The list is comprehensive, in that its coverage is broad enough to cover most modern day application development, and the base standards selected have been determined to be consistent with POSIX.1 {2}.

While this guide does not list all base standards, it is still a large list, and in fact the list contains base standards that might not be consistent with each other (choose any two standards from the POSIX OSE and they might not be consistent with each other.) The process of profile writing addresses this.

The profile writer reduces even further the list of base standards to just the (relatively) few that are needed to provide portability and interoperability in a given functional area. In the process, the profile writer grapples with the coherence of the selected base standards by choosing only those that will work together to get the particular job done. Profile writers should also deal with *harmonization*,[3] which means making the profiles consistent with each other where they overlap. This can often be done among profiles even where the functional areas served differ greatly. Procurements specifying two profiles that have been harmonized by their authors have the benefit of knowing that the two will not conflict with each other.

By specifying compliance to a particular profile in a procurement, a consumer easily references a set of multiple base standards that have been determined to: serve a particular purpose and work together.

The benefits and relationships do not end here, however. Since profiles can be constructed to reference profiles as well as base standards, future profile writing will be even easier.

NOTE: An analogy is in the construction of electronic equipment such as computers. The basic building blocks are ''components,'' such as memory chips and capacitors, which can be fabricated into larger building blocks such as printed circuit boards, which can be fabricated (with other components or printed circuit boards) into larger building blocks, such as standalone computers, which can be fabricated into larger building blocks such as department wide networks of computers, etc. Likewise, a few base standards (the basic building blocks), can be gathered together into ''component'' profiles, which can then be gathered together (with other base standards or component profiles) into larger ''platform'' profiles, which can be gathered together into larger ''application area'' profiles. (See 6.3.3.5.)

The development of profiles from the primary building blocks (base standards) results in larger building blocks (profiles) that can then be incorporated into future profiles and also into future versions of this guide.

The Importance Of Profiles

Profiles are important for a number of reasons:

- Profiles select one or more base standards or profiles and specify options and parameters within these. This provides a clear statement of specifications that describe the standards for the target functional objective(s).
- Profiles include information about the relationship between the standards included (i.e., coherency is an objective).
- Profiles are a clear method of communication about the specific standards needed for an application domain and can be used in procurement, in conformance testing, and as a target for applications development.

---

3) This should not be confused with *international harmonization*, which refers to a specific process that must be followed in the approval process for International Standardized Profiles (ISPs).

6.3 Guidance to Profile Writers

Responsibility: Bob Gambrel

This clause expands the concept of profiling in the manner needed by profile writers and provides detailed guidance to those writers. It includes a description of the basis for this guidance, expands on the purposes served by profiles, and finishes with more detailed guidance specifically aimed at those writing profiles.

Using this guide as a basis, profile writers can develop their own informal profiles, suited to their own needs, or formal standards bodies can develop formal, balloted profiles. This clause details the requirements that should be met by developers of profiles whether they are POSIX SPs, standardized profiles, or less formal profiles. Standardized profiles are formal profiles that meet the requirements of a sponsoring standards body. Standardized profiles that also meet the requirements for POSIX-based profiles (rules established by IEEE) are called POSIX standardized profiles (POSIX SPs.) For more information about writing POSIX SPs, see Annex A.

Note to reviewers: Annex A has important information in relation to this section that should be reviewed.

6.3.1 Basis for This Guidance

Many of the ideas and concepts for profiling described in this section derive from the work of ISO/IEC JTC 1 SGFS as documented in ISO/IEC TR 10000-1. Some items specified in that document that are not covered here include:

- International standardization considerations
- Conformance issues
- Processes and procedures
- Maintenance
- Taxonomy

Additionally, some consideration was given in this guidance above and beyond that given in ISO/IEC TR 10000:

- Standardized profiles and POSIX standardized profiles as a conceptual extension to International Standardized Profiles (ISP).
- IEEE basis, not ISO basis, for formatting rules; see Annex A.

Writers of profiles following the guidance of this clause should refer to Annex A if they intend to propose IEEE acceptance as a POSIX SP and to ISO/IEC TR 10000 if they intend to propose acceptance as an ISP.

6.3.2 Purpose of Profiles

Profiles define combinations of base standards and profiles for the purpose of:

- — Identifying the base standards, together with appropriate classes, subsets, options, and parameters, that are necessary to accomplish identified functions for purposes such as interoperability and portability.
- — Providing a system of referencing the various uses of base standards that is meaningful to both users and suppliers
- — Enhancing the availability for procurement of consistent implementations of functionally defined groups of base standards that are expected to be the major components of real application systems
- — Promoting uniformity in the development of conformance tests for systems that implement the functions associated with the profiles

6.3.3 Detailed Guidance to Profile Writers

6.3.3.1 The Relationship to Base Standards

Base standards specify procedures and formats that facilitate application portability and interoperability. They provide options, anticipating the needs of a variety of applications and taking into account different capabilities of real systems and networks.

Profiles further promote portability and interoperability by defining how to use a combination of base standards for a given function or application area. Profiles, by definition, do not define new application interfaces.

In addition to the selection of base standards, a choice may be made of permitted options for each base standard and of suitable values for parameters left unspecified in the base standard.

Profiles should not contradict base standards, but should make specific choices where options and ranges of values are available. Profiles must include all of the items made ‘‘mandatory’’ by the standard. The choice of the base standard options should be restricted so as to maximize the probability of interworking between systems implementing different selections of such profile options, consistent with achieving the objectives of the profile.

A profile makes explicit the relationships between a set of base standards used together (relationships that are implicit in the definitions of the Base Documents themselves) and may also specify particular details of each base standard being used.

A profile may contain conformance requirements that are more specific and limited in scope than those of the base standards to which it refers. While the capabilities and behavior specified in a profile will always be valid in terms of the Base Documents, a profile may exclude some valid optional capabilities and optional behavior permitted in those base standards.

Thus, conformance to a profile implies, by definition, conformance to the set of base standards that it references. However, conformance to that set of Base Documents does not necessarily imply conformance to the profile.

6.3.3.2 Main Elements of a Profile Definition Document

The definition of a profile should comprise the following elements:

- A concise definition of the scope of the function for which the profile is created and of its purpose
- Reference to a set of base standards and other profiles, including precise identification of the actual texts of the base standards and profiles being used and of any approved amendments and technical errata, conformance to which is identified as potentially having an impact on achieving portability and interoperation using the profile
- Specifications of the application of each referenced base standard and profile, covering recommendations on the choice of classes or subsets and on the selection of options, ranges of parameter values, etc.
- A statement defining the requirements to be observed by systems claiming conformance to this profile, including any remaining permitted options of the referenced base standards and profiles, which thus become options of this profile

Systems that interoperate can perform different but complementary roles (e.g., an initiator-responder or a master-slave relationship). In such a situation the profile should identify the separate roles that may be adopted by a system, and these should be stated as either mandatory requirements or options of the profile, as appropriate.

6.3.3.3 Profile Objectives

Completeness

A profile should be complete with respect to its functionality objectives. This may well be an iterative process, since the understanding of the requirements and standards will evolve. Completeness means that all areas where standards should be applied have been identified and the requirements defined. Where standards exist, they have been included, and the options within those standards have been addressed. Where standards do not exist, but are needed, this has been documented in the profile.

It may be appropriate to document (probably in a nonnormative appendix) specifications and alternatives available in areas where standards have not been defined. The meaning of this concept will be relative to the forum for acceptance of the profile. If the profile is targeted at ISO acceptance, then ISO DIS and IS standards should be the reference point, where as a US Government profile might be focused on FIPS and ANSI standards. Within private industry, consortium and even vendor specific specifications could be incorporated, keeping these as examples and not explicit requirements, which will simplify harmonization with formal standards as they emerge. Where standardized profiles are being developed and gaps are identified, the profile writer should identify the requirements that are not satisfied by a standard. If there is a preliminary specification available that addresses many of the requirements, that specification should be referred to informatively.

Clear Communications

A key objective for the profile is clear communications between the affected parties. Users, software developers, and platform suppliers all need to have the same terms and specifications. The application software developers and system vendors need a common set of specifications to target for their development efforts.

Harmonization

Harmonization [4] means making the profiles consistent with each other where they overlap. This can often be done among profiles even where the functional areas served differ greatly. This assures that the maximum practical agreement exists between different profiles, maximizing the implementations of that common ground.

Validation

A profile addresses validation in two different ways.

Firstly, by selecting options and parameters within the profile, validation is potentially made simpler.

Secondly, by including more than one base standard, validation potentially becomes more difficult. Now validation extends beyond just insuring a single standard is being complied with into the area of insuring that the interactions between and among multiple base standards is also being complied with.

Coherence

The simple selection of a group of standards does not assure that they will work together on a platform in a predictable way. A profile should contain a matrix of all standard components compared to each other and state what relationship exists between them. A profile may be coherent if it states that between two standards no relationship needs to exist, that none shall exist, or that a specified relation shall exist. Not to speak to an intersection in the matrix would indicate that the issue of coherence has not been addressed.

4) Refer to the earlier footnote on *international harmonization*. E

Gap Identification

In the process of developing profiles, there may be gaps in coverage by standards that become apparent. These may exist in terms of the characteristics available with one standard that need to be made available from another, or missing standards, or additional functionality that is needed for a specific applications activity. So, an additional objective for a profile effort is to document the requirements for such additional work and forward it to the appropriate standards effort. Profile groups in industry should consider providing expertise to the associated standards groups to assure that the resulting standards meet the needs of that applications area.

6.3.3.4 Methods for Developing Profiles E

To Be Determined . E

6.3.3.5 Types of Profiles

Three different types of profiles have been, or are being, defined by the procedures described above:

— Component Profiles

— Application Area Profiles

— Platform Profiles

A Component profile is mostly a subset of a single standard. The profile developers specify mandatory options for a specific domain, options that are not desirable for that domain, gaps in that parent standard, and, if necessary, specifications to fill that gap. Examples of such profiles are MAP, TOP, and GOSIP profiles and possibly the POSIX. 13 embedded realtime POSIX profile if it continues to be based exclusively on functions chosen from the POSIX. 4 realtime standard.

An Application Area Profile is created from multiple standards that specify multiple, diverse types of functionality needed for a particular application area (e.g., database, networking, graphics, operating system). The application area profile developers specify all the diverse standards necessary for the application area in question. Within each standard, they identify mandatory options, functions and options that are not needed, gaps in the standards, and, if necessary, specifications to fill the gaps. Examples of application area profiles are the POSIX. 10 supercomputing and POSIX. 11 transaction processing profiles.

A Platform Profile focuses on the functionality and interfaces needed for a particular type of platform. The platforms could be traditional platforms (such as time sharing systems) or relatively new or emerging platforms (e.g., workstations, personal computers, or symmetric multiprocessing systems). A platform profile could be created from one or multiple diverse standards. As with other types of profiles, the profile developers have to specify the standards, options, standards gaps, and if necessary, specifications to fill the gaps. Examples of platform profiles are the POSIX. 18 Platform Profile for Traditional Multiuser UNIX systems and the

POSIX. 14 Multiprocessing profile.

All three types of profiles can be seen in the next section.

Section 7: POSIX SP Profiling Efforts

Responsibility: Wendy Rauch

7.1 Introduction

This section maintains the list of currently known POSIX Standardized Profiles (POSIX SPs). This list is a factual record of which POSIX SPs exist, or are in preparation, together with a summary description of the scope, scenario, and model for each profile. These POSIX SPs might be useful as building blocks for other profiles.

7.1.1 Approved POSIX Standardized Profiles

There are currently no approved POSIX SPs.

7.1.2 POSIX Standardized Profiles In-Progress

The current efforts to develop POSIX SPs are summarized in Table 7-1.

7.2 General Purpose POSIX SPs

7.2.1 POSIX Platform Environment Profile

7.2.1.1 Rationale and Overview

The POSIX Platform Environment Profile, IEEE POSIX.18, is a platform profile based on POSIX.1 {2} and related standards. It defines the functionality and standards needed for a system that is as similar as possible to the traditional UNIX operating system's interactive, multiuser development and run-time environment.

The platform profile is valuable for many users, vendors, programmers, and procurement officers who do not have the time or desire to analyze and specify all the individual interfaces for a system they need. The platform profile obviates this analysis by enabling the users to point to a single document that specifies exactly what they should order to obtain a system that looks like traditional UNIX systems, except that the POSIX platform profile will be totally based on formal

Table 7-1 – POSIX SPs In Progress

Project Name	Taxonomy	Profile Name	Profile Type
IEEE P1003.10		Supercomputing	Application area profile
IEEE P1003.11		Transaction Processing	Application area profile
IEEE P1003.13		Realtime, Multipurpose Systems	Application area profile
IEEE P1003.13		Realtime Embedded Control System	Application area profile
IEEE P1003.13		Realtime Intermediate	Application area profile
IEEE P1003.14		Multiprocessing Application Support	Platform profile
IEEE P1003.18	USI-P001	POSIX Platform Environment Profile	Platform profile

NOTES:

(1) At this time it is not known whether the three realtime profiles will be contained within a single multipart POSIX SP, or separate single-part POSIX SPs.

(2) While the issue of a taxonomy for POSIX SPs has not been decided, a placeholder has been provided and a proposed taxonomical name for one profile has been listed.

standards.

7.2.1.2 Content of the Platform Environment Profile

The POSIX Platform Environment Profile consists of:

- ISO/IEC 9945-1, with a selection of options and definitions of parameters;
- All of the POSIX.2 (Shell and Utilities) and, optionally, POSIX.2a (User Portability Extension); and
- At least one of the following languages: ISO C, Ada, or FORTRAN.

To reflect the goals and intent of the POSIX.18 working group, the POSIX platform profile document also commits to specifying additional specifications in the future, when those specifications are completed and approved as standards. These specifications include system administration, secure/trusted systems extensions, realtime facilities, verification testing facilities, Ada and FORTRAN language bindings, graphical user interfaces, and network interface facilities.

The POSIX platform profile is expected to be the pioneer Application Environment Profile submitted to ISO for international approval. The concept of Application Environment Profiles and Platform Profiles is new. How ISO handles the international standardization of the POSIX platform profile, and the profile issues resolved, will likely set a precedent followed in the development of other profile standards.

7.2.2 Multiprocessing Systems Platform Profiles

7.2.2.1 Rationale and Overview

The POSIX Multiprocessing Systems Profile (IEEE POSIX.14) is a platform profile. Like the POSIX PEP (POSIX.18), the Multiprocessing Systems profile defines the functionality, standards, and options within standards that are needed for development and execution on a multiprocessing platform.

The Multiprocessing Systems profile is intended for use by multiprocessor vendors, application developers, users, and system administrators. It is important because it is designed to support portability of multiprocessing applications, as well as users and system administrators in multiprocessing environments.

The Multiprocessing Systems Profile has two major goals. The first one is to make POSIX safe for multiprocessing. This goal requires the POSIX.14 working group to identify and address the caveats, problems, and failings of POSIX base standards for multiprocessing platforms. Examples of these failings range from reentrant-function problems to potential problems with threads.

The second goal is to make POSIX useful for multiprocessing. This goal requires the POSIX.14 working group to ensure that POSIX supports the functionality needed by multiprocessing platforms. An example of this is ensuring that POSIX has capabilities to allow vendors to parallelize software functions. In the absence of parallelizing standards, the details of what happens when the same software functions are used on different multiprocessor system vary.

7.2.2.2 Content of the Multiprocessing Systems Profile

The Multiprocessing Systems platform profile identifies standards, options, and gaps in the standards relevant to multiprocessing. It also identifies additional requirements not satisfied by existing standards and, in an informative annex, suggests interfaces to extended services that can satisfy some of these requirements. In addition, the POSIX.14 Multiprocessing Systems Group will propose changes and amendments to a variety of relevant standards in order to encourage the specifiers of these standards to add functions and options that accommodate multiprocessing requirements.

Standards particularly relevant to the Multiprocessing System Profile include the POSIX Pthreads extension (IEEE POSIX.4a), the supercomputing batch scheduling standard (IEEE POSIX.15), and the supercomputing proposed checkpoint and restart facilities (IEEE POSIX.10). Since checkpoint and restart facilities will be added to the POSIX.1 {2} standard, POSIX.1 {2} is also of concern to the Multiprocessing Profile.

The Multiprocessing Systems profile will specify both general-purpose-computing and multiprocessor-specific standards. General-purpose standards planned or under consideration for the Multiprocessing Systems profile include:

— The IEEE POSIX.1 core POSIX system, POSIX.2 POSIX Shell and Utilities, and POSIX.2a User Portability Extension;

— The IEEE POSIX.4 realtime extension;

— The IEEE POSIX.4a: POSIX Pthreads extension;

— The IEEE POSIX.6 POSIX security standard and POSIX.7 system administration standard;

— The Ada language bindings (IEEE POSIX.5) and FORTRAN language bindings (IEEE POSIX.9) to POSIX;

— The IEEE POSIX.10 Supercomputing Profile, POSIX.11 Transaction Processing Profile, and POSIX.13 Realtime Applications Profiles.

As other standards emerge, they too will be incorporated in the Multiprocessing Systems profile. An annex of this document will deal with, and list, relevant emerging standards to provide an idea of the Multiprocessing Systems profile's direction.

Multiprocessing-specific requirements identified by the POSIX.14 Multiprocessing working group include:

— System administration tools for multiprocessors;

— Parallelizing compilers;

— Explicit parallelism;

— Threads;

— Thread-safe libraries;

— Message-passing IPC;

— Parallel utilities (e.g., `find`, `grep`, `make`, etc.);

— Scheduler controls;

— Processor allocation: mandatory/advisory;

— Processor binding;

— Degree of symmetry: I/O, computation, memory.

Standards will be needed for many of these requirements. Many of these requirements will, therefore, become the subject of a POSIX.14 working group proposal for a new standardized function or an option in other standards.

7.2.3 Supercomputing

7.2.3.1 Rationale and Overview

The Supercomputing Application Environment Profile (IEEE POSIX.10) is a profile designed to support application and programmer portability in POSIX-based supercomputer environments. The profile's goal is to allow supercomputer application code to be ported to other sites, reduce the learning curve of users, and encourage production of timely third-party applications.

The need exists for such a profile because of the differences between supercomputing environments and traditional application environments. One difference is that supercomputing jobs are computationally intensive, very long running, and very demanding of resources. Another is that the cost of the supercomputer CPU and many of its peripheral resources is extremely high.

Ordinary POSIX standards are not applicable in their entirety to supercomputer environments because the traditional UNIX-based POSIX functions are not adequate to meaningfully manage the use of, and accounting for, a supercomputer or its resources. Furthermore, supercomputers need much better tape handling, multiprocessing, and other capabilities than POSIX or UNIX specifications presently support.

7.2.3.2 Content of the Supercomputing Profile

The Supercomputing Application Environment Profile identifies POSIX base standards and other relevant standards that support supercomputing requirements. Where none exist, the POSIX.10 working group will define the functionality itself, or instigate the formation of a new group to define it. In addition, the POSIX.10 working group is taking some of the traditional modifications built to allow UNIX systems to run on supercomputers, and making those modifications both consistent across supercomputers and portable to users, system administrators, and applications.

Base computing standards specified by the supercomputing profile (or planned for specification when the standards are completed) include:

— The IEEE POSIX.1 {2} core POSIX system, POSIX.2 POSIX Shell and Tools, and POSIX.2a User Portability Extensions (and the corresponding FIPS standards);

— The IEEE POSIX.4 realtime work (particularly the use of its asynchronous I/O facility);

— The IEEE POSIX.6 POSIX security standard and POSIX.7 system administration standard;

— Several graphics standards, including ISO GKS, PHIGS, and CGM, ANSI IGES, and the X Consortium's PEX.

— X3H2.6 (also called X11) for windowing;

— Several programming languages, including ISO, ANSI, and the NIST's FIPS for C, FORTRAN-77, Pascal, Ada, Common LISP, and COBOL.

— TCP/IP protocol stacks and network applications (e.g., file transfer and messaging) now and OSI in the long-term;

— The IEEE POSIX.8 Transparent File Access standard for distributed file management;

— The X3T5.5 Remote Procedure Call (RPC).

The nonstandardized and nonavailable supercomputing functions identified in the POSIX.10 profile include:

— Batch system scheduling, administration, and network definition;

— Checkpoint recovery;

— A resource manager;

— A better tape management facility;

— Better mass storage/archiving facilities.

There are no existing standards for batch scheduling and administration facilities. Batch scheduling and administration extensions to POSIX base standards are currently being defined by the IEEE POSIX.15 working group—a group spawned by the Supercomputing profile working group.

To meet recovery and archiving requirements, the POSIX.10 working group defined system interfaces for functions that perform ''checkpoint,'' ''restart,'' and better magnetic tape handling (e.g., to rewind a tape under program control). These interfaces have been submitted to the POSIX.1 working group for inclusion in the next POSIX.1 {2} revision.

7.2.4 Transaction Processing

7.2.4.1 Rationale and Overview

The Transaction Processing Application Environment Profile (IEEE 1003.11) is intended to support the development of portable online transaction processing (OLTP) applications in POSIX environments. This profile is targeted at application developers and open system services suppliers. It is important because transaction processing is a major area of business for most large computer vendors and it plays a major role in the daily operations of most users. There are currently no existing POSIX functions that specifically address OLTP needs.

7.2.4.2 Content of the Transaction Processing Profile

The Transaction Processing profile's goal is to identify the interfaces and standards relevant to OLTP, and optional functions in existing standards that must be made mandatory for OLTP applications. The profile will specify general-purpose standards, as well as standards unique to OLTP.

The Transaction Processing Profile's specifications include or plan the following generic and transaction processing-specific standards:

— The ISO/IEC 9945-1: 1990 (POSIX 1003.1) core POSIX system interfaces (including required options, minimum values for certain variables, and particular environment variables needed for OLTP applications);

— The IEEE 1003.2 Shell and Utilities' software development utilities option, C language development utilities option, and C language bindings option;

— The IEEE 1003.2 `getconf` utility;

— The realtime files and asynchronous input and output features from the IEEE 1003.4 Realtime POSIX Extensions;

— The IEEE 1003.6 POSIX security standard;

— The ISO/IEC, ANSI, and FIPS C and COBOL programming languages;

— TCP/IP networking in the short term and OSI in the long-term;

— The X3T5.5 Remote Procedure Call (RPC)

— The ISO SQL database language;

— The ISO Distributed Transaction Processing 10026.1, .2, and .3 for communication of transaction information.

The Transaction Processing profile also identifies extensions needed to existing standards to support distributed transaction processing. Important extensions that need to be defined include those related to the two-phase commit, as well as others related to making RPCs robust.

The P1003.11 working group is working with the ISO RPC Group to add transaction semantics to the Networking working group's RPC specifications. These extensions will be incorporated in the Transaction Processing profile. Plans are also for the 1003.11 profile to draw on the transaction processing work being produced by the X/Open consortium, particularly on the XA interfaces (the interface between a Transaction Manager and a Resource Manager).

7.2.5 Realtime Application Profiles

7.2.5.1 Rationale and Overview

Different types of realtime applications have different characteristics and diverse requirements. For example, embedded systems generally do not need the full functionality of an operating system, nor do they require all the IEEE POSIX.4 realtime extensions. Compliance with the entire realtime standard and/or POSIX operating system interfaces could reduce the embedded system's responsiveness and increase the amount of memory needed for systems that need to be embedded in limited space. High-end realtime systems, on the other hand, have softer realtime requirements. However, they need the full operating system and realtime functionality.

Therefore, the POSIX.13 working group was formed to define profiles for various types of realtime applications. The realtime profiles defined will determine which interfaces must be implemented for a given type of realtime system to claim conformance to the realtime standard.

7.2.5.2 Targeted Realtime Application Profiles

The POSIX.13 working group is defining profiles to address several types of realtime applications. These include:

- Low-end, embedded systems (often known as ''hard'' realtime systems);
- Mid-range realtime systems with medium-level critical realtime constraints;
- High-end realtime systems.

7.2.5.2.1 Embedded Realtime Systems

Embedded realtime systems are typically standalone systems used for robot controllers, automated systems controllers, instrumentation, high-speed data acquisition, satellite subsystem control, flight control, some process control, and some testing. Time-critical responsiveness is a key requirement of embedded systems. In the absence of a standard, the realtime functionality required for embedded systems is generally provided by a proprietary realtime kernel or a simple home-grown monitor using memory mapped I/O.

Since low-end embedded systems need only minimal functionality, the POSIX.13 working group will select a relatively small number of POSIX.4 and POSIX.1 {2} functions that will be required for portable realtime embedded systems. These functions will be selected for several types of embedded applications.

One type of embedded application is a minimal system, usually buried deeply in the overall system electronics. Such minimal applications have no requirements for a file system, multiple processes, or I/O via specific device drivers. The minimal realtime profile, however, will specify the POSIX.4a threads extension to support multiple flows of control.

The second type of embedded application is often used in control systems. Realtime controller applications require a file system and threads, but not multiple processes.

7.2.5.2.2 Mid-Range Realtime Applications

Mid-range or intermediate-level realtime profiles are targeted at compute-oriented applications that are typically used in avionics, radar systems, submarines, and medical imaging equipment, as well as controllers that control a group of robots or a subsystem on the factory floor. These applications tend to run on platforms that are dedicated to a single application set or mission mode.

The design complexity of such dedicated realtime applications varies from simple to complex to accommodate a range of requirements. Such requirements may include sophisticated signal processing capabilities, but do not necessarily include a file system. A profile that satisfies these requirements would likely specify most of the POSIX.4 functionality (except for file system facilities), along with relevant options from the POSIX.4 and POSIX.1 {2} standards and the POSIX.4a threads extension.

7.2.5.2.3 High-End Realtime Applications

High-end realtime applications are applicable to complex, multipurpose realtime systems. Such multipurpose realtime systems typically are used in military command and control, in space station control systems, in systems that control robot or factory subsystems, as the operating system for high-end simulation systems, and at high-functionality realtime application that are paced by operator interaction.

The current realtime, multipurpose profile is geared to full-function realtime systems such as simulation applications and embodies most of the existing practice in the simulator world. Since simulation systems have a greater design complexity than embedded or mid-range systems, and need much greater functionality, the multipurpose realtime profile will most likely require all or most of the POSIX.4 and POSIX.1 {2} standards. This profile does not require threads. It does, however, specify the X11 window system as the basis for a human-computer interface.

Annex A
(informative)

Considerations for Developers of POSIX SPs

A.1 Introduction

Responsibility: Bob Gambrel

The contents of this Annex are illustrative of rules that might be developed for the submitters of POSIX Standardized Profiles (SPs).

This Annex contains modifications and comments relating to the use of the *TCOS-SSC POSIX Standards Style Guide* {B6} in POSIX SPs.

A.2 Scope

While Section 6 addressed profiles generally, this Annex addresses considerations for developers of formal POSIX Standardized Profiles. It builds directly upon the concepts, principles, and guidance of Section 6.

Note to reviewers: This Annex is not complete, in that more work is required in the domain of POSIX profiles.

Future work in the area of profiling will be done by IEEE and the standards community. This document, and the guidance it provides, will be updated as appropriate. The major areas expected to be addressed are:

- *International standardization considerations*
- *Conformance issues*
- *Taxonomy of POSIX SPs*
- *Registration of POSIX SPs*
- *Delegation of authority to call something a POSIX SP (Note: Currently, this document does not prohibit another group beside IEEE from calling their document a POSIX SP.)*
- *Clarification of base standards referencing issues such as subsetting and the handling of options*
- *Editorial issues such as guidance on the correct level of detail*

— *Additional guidance on referencing base standards and ''standards in progress''*

A.3 The Role of POSIX SPs

In 6.3.3.5, a classification scheme was given for profiles in which three different ''types'' were identified. That scheme is based, essentially, on the scope covered by the profile. Another useful classification scheme, based on scope and on who develops the profiles, is presented in this annex.

Figure A-1 shows these classes of profiles and the relationships between them and base standards.

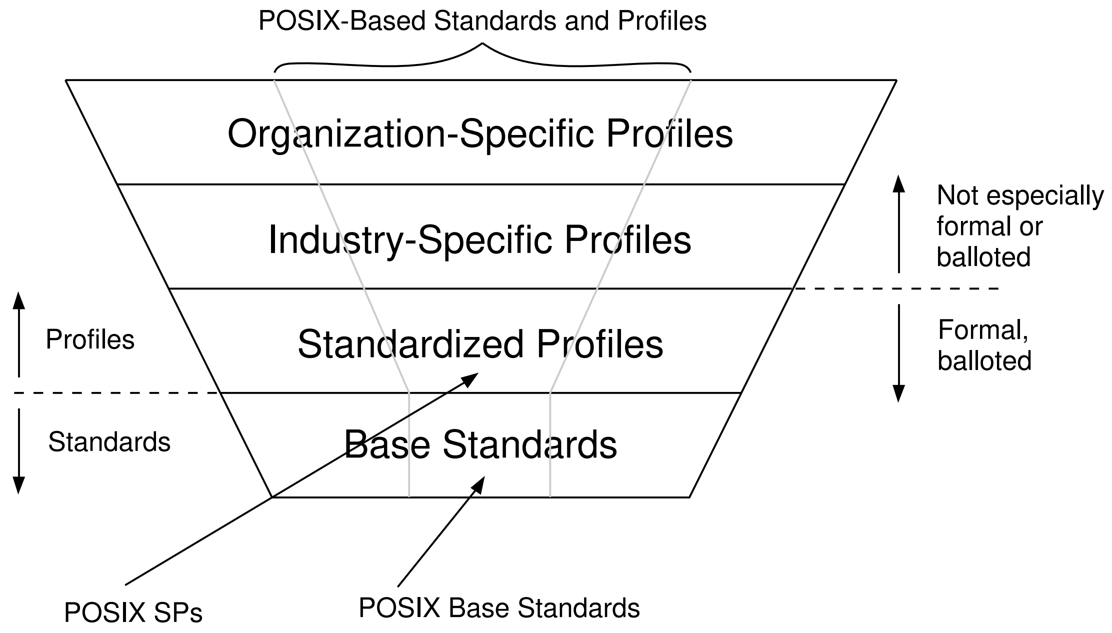

Figure A-1 – Universe of Profiles and Standards

Base standards cover a universe of diverse needs. POSIX base standards (e.g., POSIX.1 {2}, P1003.4, . . .) cover a narrower set of needs related to 'POSIX.'' In the figure, the POSIX base standards are shown as a small subset of the larger world of base standards.

At the other end of the spectrum, organization-specific (e.g., company-specific) profiles are large in number and range even more widely in their coverage. (There are many more organizations procuring systems, and effectively writing profiles, than there are committees writing standards.)

Industry-specific profiles are based on specific industry needs. From the point of view of the organization-specific profile writer, industry specific profiles are applicable to many organizations (in the same industry), and hence are possibly not precisely what any specific individual organization needs. They address the broad consensus of the industry, from which there is usually deviation when you look at individual organizations whose needs range further.

Standardized Profiles are formal balloted documents. POSIX SPs are the subset of standardized profiles that pertain to the POSIX base standards. While not limited to just POSIX base standards, POSIX SPs nonetheless provide a distinctly POSIX-oriented view of the base standards.

An organization wishing to procure a ''POSIX'' based system, then, could first develop its own organization-specific profile, which it could base on POSIX-oriented industry-specific profiles (if available), which in turn could be based on POSIX SPs, which of course are based on the various POSIX base standards.

POSIX SPs provide an industry-neutral building block for creating industry specific profiles. The developers of POSIX SPs do not have to have knowledge of any particular industry. They furthermore help ensure coherence among the many base standards referenced, particularly among the various POSIX base standards. As such, probably, most POSIX SPs will be created by the IEEE POSIX working groups meeting concurrently with IEEE POSIX base standards working groups. Meeting concurrently at the same place helps ensure the coherence of the base standards and the harmony among the POSIX SPs.

A.4 Special Rules for POSIX SPs

While no rules have yet been developed by IEEE for POSIX SPs, the remainder of this annex gives examples of what such rules might say and identifies some issues for which rules might be drafted.

The following criteria for calling a profile a POSIX SP were developed according to some general principles that have the aim of giving definite value to the word ''POSIX'' when used with regards to profiles. The general principles are:

(1) There is minimum content. Specifically, a POSIX SP must reference some part of the suite of POSIX base standards. (Which part specifically is contentious.)

(2) The POSIX SP must follow a specific approach to conformance (specifically the P1003.3.1 test methodology.)

(3) The POSIX SP must adhere to the POSIX Reference Model.

(4) There is maximum content; i.e., some consideration must be given to how the POSIX SP goes beyond the POSIX OSE as described in this guide.

(5) Exceptions to the previous principles are expected, requiring a rule-making and enforcement body to make those exception decisions.

POSIX SPs are Standardized Profiles that are related to ''POSIX.'' This subclause specifies the rules that need to be followed that distinguish POSIX SPs from ''Non-POSIX SPs''.

Each POSIX SP is based on, and shall include, one of the following two base standards sets:

(1) POSIX.1 {2} or POSIX.2 (as verified by the P1003.3 methodology), or

(2) A particular subset of POSIX.1 {2} and P1003.4 that is being specified for a Minimal Realtime profile (as verified by the P1003.3 methodology.)

Additionally, each POSIX SP adheres to the structure defined by the POSIX OSE reference model.

An approved POSIX SP shall make reference only to base standards identified in this guide (1003.0) as being part of the POSIX OSE. Two specific exceptions to this general rule are allowed for as described here:

(1) Reference can be made to required base standards that are clearly outside of the scope of the POSIX OSE. Examples of the functionality that may require the use of this expedient are:

— Physical connectors

— Electrical characteristics

— Safety requirements

Such reference to items outside the scope of the POSIX OSE shall be justified on a case-by-case basis. It shall be accompanied by details of the body responsible for the distribution and maintenance of the referenced base standard.

(2) Reference can be made to required base standards that are being proposed for inclusion in a future version of the guide. Examples of this would be specification of a later version of a base standard that is already included within the POSIX OSE, or of an additional programming language base standard, not yet included within the POSIX OSE.

In such cases, the POSIX SP should be identified as a POSIX Preliminary SP and the specific references should be clearly noted and justified on a case by case basis.

A POSIX Preliminary Standardized Profile (POSIX Preliminary SP) is a POSIX SP that satisfies all requirements of a POSIX SP except that it is not a subset of the POSIX OSE. [It therefore contains at least one standard or profile that is outside the POSIX OSE. It is expected that application would be made to POSIX.0 to include the standard(s) or profile(s) in the POSIX OSE.]

A further restriction of POSIX SPs is the necessity to (normatively) reference only standards that are recognized by the IEEE. This is limited to IEEE and ISO standards.

Approval of a POSIX SP shall not change the status of any documents referenced by it.

The development of a POSIX SP may indicate the need to modify or to add to the requirements specified in a base standard. In this case, it is necessary for the POSIX SP developer to liaise with the body responsible for that base standard so that the required changes may be made through established methods such as defect reporting, amendment procedures, or the introduction of new work.

A.5 Other Issues

A significant number of issues remain to be addressed concerning the management of POSIX SP development. Some of the issues and the concerns are summarized here.

Coherence

The insurance of coherence among the many base standards referenced by a profile has been found by profile writers to be an onerous task. The profile writer's burden could be eased significantly if base standards writers address coherence at the outset. Specifically, all the P1003.x base standards should be developed to maximize their coherence. This is seen as a management issue for TCOS-SEC, the sponsoring body of the P1003.x standards.

Conformance

The development of conformance statements and test methods for profiles is a significant challenge for profile writers. The challenge is most acute in the area of conformance of standards that are being developed outside of P1003. A premise for the profile writing rules associated with conformance must be that the profile writers are not really experts in the referenced standards. Profile writers (especially at this early period in their development) must not be overburdened with untested conformance writing rules. A possible solution is to create a new project under the auspices of P1003.3 to actually generate new test methods and actually write the necessary assertions for the first profile. (This approach was used also for the initial POSIX base standard.)

Base Standards Working Groups

Because profile writers are in some sense the customers of base standards, it is important for base standards writers to address with priority and urgency the gaps identified in the development of POSIX SPs.

Scope and Number of POSIX SPs

How many different POSIX SPs are appropriate and how broadly ranging should be their scope? Should POSIX SPs be rather narrowly focused, spanning just a few base standards, or should they address a large number of base standards?

Issues Pertaining to Referencing Base Standards

Many practical writing issues pertain to referencing, for instance, parts of base standards. This includes not only referencing options, but even the concept of subsetting, or reducing the functionality of a base standard. Also an issue is how to reference multiple versions of the same standard (e.g., two different COBOL standards.)

POSIX SP Procedures and Rules

What does it mean to be a POSIX SP? Rule making for use of the word 'POSIX'' must address criteria for such use. Also, many issues remain to be resolved in the area of ballot procedures. Should IEEE delegate to others the ability to develop POSIX SPs? If so, should IEEE maintain a registry of such efforts?

A.6 Conformance to a POSIX SP

A POSIX SP must address test methods for itself. In the simplest case, testing the base standards referenced is sufficient. In more complex cases, additional test methods will be necessary. In the worst case (if a base standard is subsetted, for example), the test methods for the base standards may have to be rewritten or expanded within the POSIX SP.

At the same time, P1003.3 will have to consider revisions to the *Test Methods for Measuring Conformance to POSIX* to address test methods for POSIX SPs (e.g., additional assertion types, minimum requirements for testing POSIX SPs, . . .)

A.7 Structure of Documentation for POSIX SPs

This clause gives specific format and content requirements to profile writers who are developing POSIX SPs.

A.7.1 Principles

The requirements for content and format of POSIX SPs are based on the following principles:

(1) Profiles shall be directly related to base standards and conformance to profiles shall imply conformance to base standards.

(2) POSIX SPs shall follow the rules for drafting and presentation of POSIX SPs detailed here.

(3) POSIX SPs are intended to be concise documents that do not repeat the text of the base standards.

(4) Profiles making identical use of particular base documents shall be consistent, down to the level of identical wording in the POSIX SPs for identical requirements.

A.7.2 Multipart POSIX SPs

Many profiles will be documented and published as individual POSIX SPs. However, where close relationships exist between two or more profiles, a more appropriate technique can be used.

Common text between related profiles is essential to ensure consistency, portability, and interworking, to avoid unnecessary duplication of text, and to aid writers and reviewers of POSIX SPs.

A *single-part POSIX SP* shall not contain the definition of more than one profile.

The following rules apply to *multipart POSIX SPs*:

(1) A multipart POSIX SP shall contain the definition of a complete profile or of a related set of profiles.

(2) A part of a multipart POSIX SP may contain a section of the definition of one or more profiles.

(3) Where a multipart POSIX SP includes more than one profile, the part structure shall permit each profile to be the subject of a separate ballot; i.e., its constituent profiles shall be clearly identifiable, and the multipart structure shall ensure that this can be accomplished.

(4) Wherever possible, the references made from one part to another should be to complete parts. However, controlled use of one-way references to sections of other parts is permitted in order to obtain a reasonable multipart structure.

Because there may also be potential disadvantages from overuse of the multipart POSIX SP capability, such as difficulties in gaining approval for a complex linked set of parts, or reduction of the content of a part to a small amount of text, considerable care should be taken with its use.

NOTES:

(1) When a section of text appears in several profiles, possibilities exist for sharing the corresponding code (etc.) for the implementation of several profiles, and the tests applicable to the use of the referenced base standards will be applicable to the testing of several profiles.

(2) It follows that it is in the interest of the implementors to promote the identification of common sections of text as parts of POSIX SPs, but even more to promote, in future standardization and profile work, the use of already defined parts of POSIX SPs, so that profiles fall into a few ‘‘common molds.’’ In particular, this allows implementation of a part of a POSIX SP with confidence that it may be used in the implementation of profiles as yet undefined, so that products are open to future development.

(3) Possibilities exist for a complete profile to be referenced from within the definition of another profile.

A.8 Rules for Drafting and Presentation of POSIX SPs

Throughout this Annex, which is concerned with documentation content and layout, reference is made to POSIX SPs. A POSIX SP, or part thereof, may contain a whole profile definition or part of one or more profile definitions. The wording of the Annex assumes that it is describing an undivided POSIX SP that defines one profile in its entirety. Its application to the other cases is easily deduced. Note, however, that each part of a Multipart POSIX SP shall use the same format as far as appropriate.

A.8.1 General Arrangement

The elements that together form a POSIX SP are classified into three groups:

(1) Preliminary elements are those elements that identify the POSIX SP, introduce its content, and explain its background, its development, and its relationship with other standards and POSIX SPs.

(2) Normative elements are those elements setting out the provisions with which it is necessary to comply in order to be able to claim conformity with the POSIX SP.

(3) Supplementary elements are those elements that provide additional information intended to assist the understanding or use of the POSIX SP.

These groups of elements are described in the following clauses.

A.8.2 Preliminary Elements

A.8.2.1 Foreword

The foreword shall appear in every POSIX SP. It consists of a general part giving information relating to the organization responsible and a specific part giving as many of the following as are appropriate:

— An identification of the organization or committee that prepared the POSIX SP; information regarding the approval of the POSIX SP

— A statement that the POSIX SP cancels or replaces other documents in whole or in part

— A statement of significant technical changes from the previous edition

— A statement of which annexes are normative and which are informative

A.8.2.2 Introduction

The introduction shall appear in every POSIX SP. It gives specific information about the process used to draft the POSIX SP and about the degree of international harmonization that it has received.

A.8.3 General Normative Elements

A.8.3.1 Title

The title shall be composed of the following three elements:

(1) An introductory element: *Standard for Information Technology*

(2) An identification element: *POSIX Standardized Profile*

(3) A main element indicating the subject matter of the POSIX SP. For a Multipart POSIX SP, this element shall be subdivided into a general title element common to all parts, and a specific title element for each part; where necessary, this specific element may include the identifier of an individual profile. The first word of this element should be the word ''POSIX''.

Example:

Standard for Information Technology —
POSIX Standardized Profile —
POSIX Transaction Processing

A.8.3.2 Scope

This element contains two subclauses as follows:

(1) General

This element shall appear at the beginning of the POSIX SP or POSIX SP part to define without ambiguity the purpose and subject matter of the document, thereby indicating the limits of its applicability. It shall not contain requirements.

(2) Scenario

If the POSIX SP or POSIX SP part defines a profile, it shall include (where appropriate) the ''scenario'' of the profile; i.e., an illustration of the environment within which it is applicable. This may show in a simplified graphic form how this fits within the POSIX Reference Model.

A profile should first introduce the functional area being addressed and the applications activities within that area. The requirements that have been addressed should be delineated, as well as those areas outside of the scope of the profile.

A.8.3.3 Normative References

This element shall give a list of normative documents (base standards), with their titles and publication dates, to which reference is made in the text in such a way as to make them indispensable for the application of the POSIX SP. Where published amendments or technical errata to base standards are relevant to the definition of the profile in such a way as to have a potential impact on interworking or portability, they shall be explicitly referenced here.

Reference shall also be made to this guide.

A.8.4 Technical Normative Elements

A.8.4.1 Requirements

This element includes clauses relating to the use made of each of the main base standards referenced in the profile definition. The content and layout of these clauses are not defined, but can be tailored to the type of material that has to be specified in each case.

The information given shall not repeat the text of the base standards, but shall define the choices made in the profile of classes, subsets, options and ranges of parameter values. It shall be in the form of conformance requirements and may, where appropriate, be given in tabular form.

See 6.3.3 for more detail concerning the nature of the content required in this element of a POSIX SP.

A.8.4.2 Normative Annexes

Normative annexes are integral sections of the POSIX SP that, for reasons of convenience, are placed after all other normative elements. The fact that an annex is normative (as opposed to informative) shall be made clear by the way in which it is referred to in the text, by a statement to this effect in the foreword, and by an indication at the head of the annex itself.

A.8.5 Supplementary Elements

A.8.5.1 Informative Annexes

Informative annexes give additional information and are placed after the normative elements of a POSIX SP. They shall not contain requirements. The fact that an annex is informative (as opposed to normative) shall be made clear by the way in which it is referred to in the text, by a statement to this effect in the foreword, and by an indication at the head of the annex itself.

Informative annexes provide a point for documenting useful information for the users of a profile that poses no requirements. Such annexes can include:

(1) Specification of additional standards or options that will make the profile useful for specific locales (character sets, etc.)

(2) Pointers to the referenced standards and information on ordering these

(3) Pointers to related specifications that may provide additional insight or potentially serve to fill gaps in the profile

(4) Comments and concepts in using the profile for various target readers. This could include use in procurements (perhaps cross referencing

related procurement standards like the FIPS in the US). The annex may be used to provide recommendations for use that are not warranted in the standard (e.g., ‘‘Algol is not recommended for new applications development’’).

Annex B
(informative)

Bibliography

Note to reviewers: This annex is not complete. It should include references to standards, books, articles, etc., that are not required for an integral understanding of the document (as are the entries in Normative References). It currently consists only of sample entries. It will be replaced in a later draft.

{B1} ISO 7498: 1984, *Information processing systems—Open Systems Interconnection—Basic Reference Model.* [1)]

{B2} ISO 8072: 1986, *Information processing systems—Open Systems Interconnection—Transport service definition.*

{B3} ISO/IEC 8073: 1988, *Information processing systems—Open Systems Interconnection—Connection oriented transport protocol specification.* [2)]

{B4} CCITT Recommendation X.25, *Interface between data terminal equipment (DTE) and data circuit-terminating equipment (DCT) for terminals operating in the packet mode and connected to public data networks by dedicated circuit.* [3)]

{B5} CCITT Recommendation X.212, *Information processing systems—Data communication—Data link service definition for Open Systems Interconnection.*

{B5} ANSI X3.113-1987 [4)], *Information systems—Programming language— FULL BASIC.*

---

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

2) IEC documents can be obtained from the IEC office, 3, rue de Varembé, Case Postale 131, CH-1211, Genève 20, Switzerland/Suisse.

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

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

{B6} IEEE Computer Society Technical Committee on Operating Systems and Application Environments Standards Subcommittee. *TCOS-SSC POSIX Standards Style Guide.*

{B7} American Telephone and Telegraph Company. *System V Interface Definition (SVID), Issues 2 and 3.* Morristown, NJ: UNIX Press, 1986, 1989.

{B8} /usr/group Standards Committee. *1984 /usr/group Standard.* Santa Clara, CA: UniForum, 1984.

{B9} X/Open Company, Ltd. *X/Open Portability Guide, Issue 3.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

Annex C
(informative)

Standards Infrastructure Description

Responsibility: Wendy Rauch

C.1 Introduction

This annex provides a brief summary of the major national and international organizations working on the standardization of information technology.

There are two major categories of open standards organizations. One consists of formally-recognized standards bodies, responsible for definition and dissemination of public standards. Their specifications are known as formal or de jure standards. International, national, and regional standards groups, and some professional and technical organizations' standards groups are examples of formal standards bodies. Organizations specifying standards for open system usually give precedence to international standards first, then regional, national, and finally professional group standards.

The other standards organization category consists of informal bodies. Informal standards bodies are typically created by suppliers or users of information technology, usually using a consensus method, to enable the implementation of standards. They produce specifications known as industry standards or de facto standards. Certain trade associations, industry groups, vendor consortia, and user groups are examples of informal standards bodies. For informal specifications to be approved as formal standards (e.g., international or national standards) informal standards groups typically submit their specifications to formal standards organizations.

The term ''de facto standard'' is sometimes applied to popular vendor-defined systems. Such systems, however, are closed systems, often controlled in a proprietary fashion. Although they have value, closed de facto standards are not the subject of this guide.

Most standards bodies support three types of status for their standards or specifications—approved, draft, and work item. An approved standard is one that has been fully ratified by whatever means the approving standards body uses. A draft standard is one that has yet to be fully ratified, such as an ISO DIS (Draft International Standard) or a CEN ENV. Work item is a catch-all phrase for everything else, such as immature specifications, technical reports, etc., that have not yet achieved draft status.

C.1.1 International Standards Bodies Overview

Standards with the highest status are internationally agreed ones. In information technology, these are produced and published by the International Organization for Standardization (ISO). Other standards and/or recommendations are issued by the International Electrotechnical Commission (IEC), the International Telecommunication Union (ITU), and the CCITT. International standards bodies participants are normally countries and trade bodies, rather than individual suppliers or users.

C.1.2 National Standards Bodies Overview

Like the international standards bodies, most national bodies do not admit either suppliers or users directly, but receive representatives from interested trade bodies. In general, the national bodies support and adopt the international standards, developing national standards only if no international standards are available, or to meet special national requirements. Each country has a national body that is the formal representative to the international standards groups.

The relationship between the major international and national standards groups is shown in Figure C-1.

C.1.3 International and National Standards Bodies Relationship

Nongovernment standards organizations include trade associations, professional and technical societies, vendor consortia, user groups, and other special interest groups. Actual standards development occurs within these groups. The standards specified by formal standards groups within this category typically are subsequently submitted to national or international standards organizations for approval. Many informal bodies submit their specifications to formal bodies for approval as an accredited standard. (See Figure C-1).

C.2 The Formal Standards Groups

C.2.1 International and National Standards Organizations

NOTE: Only a few of the many national standards organizations are described in this subclause. E
However, the activities of these groups are representative of national standards groups in general. E

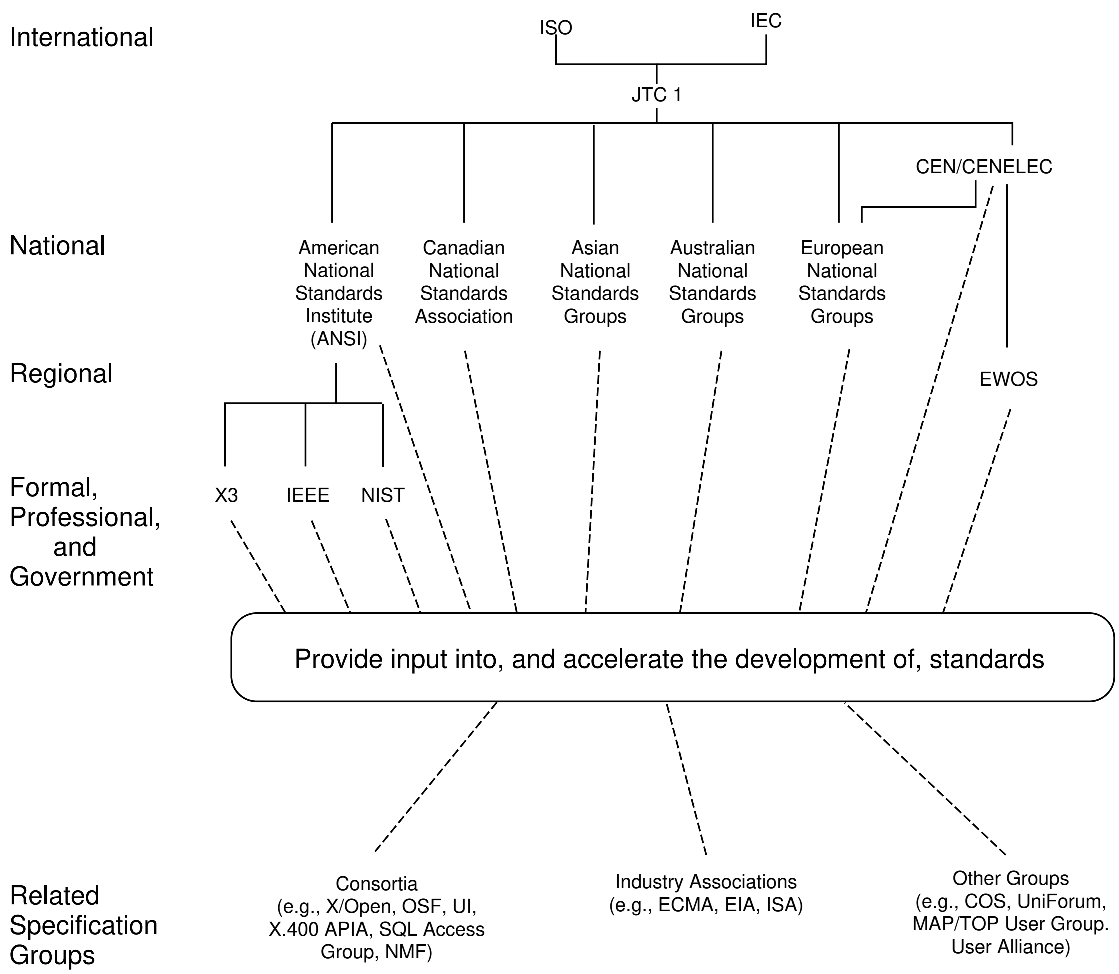

Figure C-1 – Selected Major Standards and Standards-Influencing Bodies

AFNOR : Association Francaise de Normalization

AFNOR is the French national standards body. Its responsibilities include sourcing, coordinating, approving, and promoting standards, representing the French at international meetings, and controlling the use of the NF label—a trademark that shows compliance with a French national standard. AFNOR publishes three types of standards documents— AFNOR -approved standards that are mandatory for use in the public sector, experimental standards that use new processes or techniques and whose use is voluntary, and information or guide standards.

For further information, contact Association Francaise de Normalization (AFNOR), Tour Europe – Cedex 7, 92080 Paris La Defense, Telephone: (1) 42 91 55 55, Telex: AFNOR 611 974F, Fax: (1) 42 91 56 56.

ANSI : American National Standards Institute

ANSI is the national standards coordinating and approval body for the United States. A voluntary organization founded in 1918, the ANSI performs three major types of functions.

First, the ANSI approves standards and accredits standards development groups and certification programs. ANSI does not itself develop standards. Instead, it approves voluntarily-submitted specifications that were developed by technical and professional societies, trade associations, and special interest groups, if these specifications and/or groups meet ANSI criteria for due process and consensus.

ANSI accredits three types of organizations. One is professional societies, such as the IEEE. The second is committees formed for the exclusive purpose of developing standards, such as X3. The third is accredited by ANSI to use the canvass method to develop standards. Such organizations prepare a standard using their internal procedures. Then they submit that standard to balloting by other organizations representing a variety of interests. Last, they reconcile comments and objections returned. The NIST is an organization accredited to use the canvass process for standards development.

ANSI's second major function is to represent and coordinate US interests in international, nontreaty, and nongovernmental standards bodies. ANSI's third function is to be a clearinghouse for national, international, and foreign national standards. ANSI membership is open to manufacturers, organizations, users, and communications carriers. At present, more than 220 professional and technical societies and trade associations that develop standards in the US are ANSI members, as are 1000 companies.

For further information, contact American National Standards Institute (ANSI), 1430 Broadway, New York, NY 10018, (212) 354-3300, Telex: 42 42 96 ANSI UI.

BSI : British Standards Institute

BSI is the British national standards body and is responsible for promulgation of national standards. The BSI determines the overall UK view toward international standards and conveys that back to the secretariat of the international committee.

For further information, contact British Standards Institute, 2 Park Street, London W1A2BS, United Kingdom, Telephone: 44 1 629 90 00, Fax: 44 1 629 05 06.

Canadian Standards Association (CSA)

The Canadian Standards Association (CSA), in conjunction with regulatory agencies and with the provincial and national governments of Canada, provides a single source for consensus-based standards development, conformance testing, and standards-based regulations creation. The CSA has no single counterpart in the US. Instead, the CSA handles selected functions from US testing organizations, the FCC, and ANSI.

Membership in the CSA is open to any Canadian citizen, business, or organization. Members of the CSA's technical committees developing standards are volunteers, drawn from consumers, manufacturers, government, labor, and consultants. Membership is based on expertise in the field, and not, as in the US, mainly on having a vested commercial interest. The CSA has over 900 committees handling various aspects of standards in areas such as the environment, electrical and electronics, communications and information processing, construction, energy, transportation and distribution, materials technology, and production management.

CSA programs support Canadian industry and Canadian consumers where safety and quality of merchandise sold or made in Canada are concerned. To assure product quality and safety, the CSA offers fee-based testing services. In performing such services, the CSA assumes that most manufacturers have the facilities to test their products before submitting them to the CSA for certification and approval. If they do not, the CSA provides this service. CSA certification involves the submission of the product or service by the supplier, the verification of that product or capability by the CSA, and then continued follow-up audits by the CSA to ensure that the quality of the product or service is maintained.

For further information, contact (Address and phone number TBD).

CCITT : Comite Consultatif International de Telegraphie et Telephonie

An international organization, the CCITT is part of the International Telecommunications Union, which is a United Nations treaty organization formed in 1865. It is now a specialized agency of the United Nations.

The CCITT's primary mission is to develop standards supporting the international interconnection and interoperability of telecommunications networks at interfaces with end-user systems, carriers, information and enhanced-service providers, and customer premises equipment. Every four years, the CCITT publishes the results of its work as ‘‘Recommendations.’’ Its recommendations are law where communications in Europe are nationalized.

Membership and participation in the CCITT are open to private companies; scientific and trade associations; and postal, telephone, and telegraph administrations. CCITT's principal participants are telecommunications administrations and carriers. Scientific and industrial organizations can participate as observers. The US representative is the Department of State.

For further information, contact International Consultative Committee on Telegraphy and Telephone, Central Administration Office, CH-1211, 2 rue de Varembé, Geneva, Switzerland,

CEN/CENELEC/CEPT

The Comite Europeen de Normalisation (CEN), Comite Europeen de Normalisation Electrotechnique (CENELEC), and the European Committee for Post and Telecommunications Administration are European regional standards committees responsible for developing and publishing European standards. CEN is an association of EC (European Community) and EFTA (European Free Trade Association) members. It is active in making members' standards into ISO standards and

European standards. CENELEC is the counterpart of CEN that deals exclusively with electrotechnical matters. CEPT is the CEN counterpart that deals with telecommunications matters.

CEN, CENELEC, and CEPT can be considered the European regional equivalent of ISO for two reasons. First, they have as members the national standards bodies of their eighteen EC and EFTA member states. Second, standards adopted by these organizations must be implemented in full as national standards, regardless of the way in which the member voted, and regardless of any standards that conflict with them must be withdrawn. CEN members, for example, agree to use its published standards in preference to national standards, wherever possible.

CEN, CENELEC, and CEPT were created to improve the competitiveness of European enterprise by removing technical barriers to trade and facilitating the free movement of goods within Europe. To accomplish its aims, CEN, CENELEC, and CEPT perform the following tasks:

— Create and promote European Standards (EN).

— Rapidly create prestandards (ENV) in technology areas in which there is a high level of innovation or where it is felt that future standardization requires basic guidance. ENVs are subjected to an experimental period of up to three years.

— Create harmonization documents (HD) that are more flexible than European Standards so that the technical, historical, or legal circumstances pertaining to each country can be taken into account.

— Set up a framework for European certification that supports the issuing of a European mark of conformity to certain standards and the mutual recognition of test results and inspections.

— Promote the application within Europe of ISO standards and accelerate their production.

— Work in liaison with European professional federations and numerous technical organizations to establish priority standards programs and contribute to the technical work.

For further information, contact the European Committee for Standardization (CEN), European Committee for Post and Telecommunications Administration, 2 rue Brederode, Suite 5, B-1000 Brussels, Belgium, Telephone: +322 519 6860, Telex: 26257 CENLEC.

DIN: Deutsches Institut fur Normung

DIN is the German national standards body. Its functions include those performed by the US's ANSI (e.g., developing national standards and representing Germany in international and European standards bodies such as ISO, the IEC, CEN, and CENELEC), in addition to test and certification functions that are not handled by US consensus standards organizations. Since a key DIN objective is eliminating technical barriers to free trade, DIN plays an active role in the international standards arena to ensure that German products can be used and accepted internationally.

DIN standards are not mandatory within Germany. DIN claims that it relies on the technical excellence of its standards to win converts. Further incentive for accepting DIN standards is provided because DIN standards serve as the basis for regulatory technical law in Germany. Also, without the DIN testing and inspection mark, no insurance carrier in Germany will write insurance for a product.

DIN members include groups within Germany representing manufacturers, the academic community, user groups, user organizations (e.g., consumer advocate groups), the government, and trade unions. Many DIN staff are supported by organizations or companies, rather than by DIN. DIN presently has over 20 000 standards.

For further information, contact Deutsches Institut fur Normung, Burggrafenstrasse 6, Postfach 1107, D-1000 Berlin 30, Telephone: 49 30 26 01-1, Fax: 49 30 260 12 31.

IEC : International Electrotechnical Commission

The International Electrotechnical Committee is the equivalent of ISO, but for electrotechnical standards. ISO and the IEC have converged many of their information technology efforts to form JTC 1.

For further information, contact International Electrotechnical Commission (IEC), 3, rue de Varembé, CH-1211 Geneva 20, Switzerland, Telephone: 41 22 34 01 50, Fax: 41 22 33 38 43.

ISO : International Organization for Standardization

ISO was established in its present form in 1947 with the aim of reaching international agreement on standards. A voluntary, non-United Nations treaty, ISO's membership consists of delegations from standards bodies in participating nations. ISO solicits comments from other groups as well, including ECMA, the IEEE, the NIST, and the CCITT. ISO has a close relationship with the CCITT, which is, perhaps, the most influential of all the observer groups within ISO.

ISO is responsible for the development and standardization of the Open Systems Interconnection (OSI) model. It also considers items for standardization that were developed in other standards bodies, such as ANSI. At present, for example, it is considering the core POSIX standard (P1003.1).

For further information, contact the International Organization for Standardization, Central Secretariat, 1, rue de Varembé, CH-1211, Geneva, Switzerland-40.

JISC : Japanese Industrial Standards Committee

The Japanese Industrial Standards Committee (JISC) is the national standards body of Japan. The JISC represents Japan at ISO and IEC, develops Japanese standards, and monitors and liaises with the standards-developing activities of other national organizations, especially those of the US. The goal of the JISC is to ensure that Japanese industry can compete internationally in the information technology and telecommunications industries.

The JISC has no true counterpart in other nations since the JISC has a special relationship with the Japanese government and major manufacturers. For example, the JISC's secretariat is the Agency of Industrial Science and Technology, a division of the Ministry of International Trade and Industry (MITI), which plays a central role in Japanese industry. The influence of this centralized national planning structure eliminates many areas of contention, including among companies with multinational branches, and facilitates the ability for Japanese standards groups to gain a consensus.

Major Japanese manufacturers help plan and develop standards. Foreign companies' involvement in the JISC is limited because of geographic and linguistic differences and because of restrictions on their meaningful participation. Although large-scale manufacturers may participate, user groups and small manufacturers find participation very difficult.

For information, contact Japanese Industrial Standards Committee, c/o Standards Department, Agency of Industrial Science and Technology, Ministry of International Trade and Industry, 1-3-1 Kasumigaseki, Chiyoda-ku, Telephone: 813 501 92 95/6, Fax: 81 3 580 14 18.

JTC 1: Joint Technical Committee 1

The JTC 1, established in 1987, is the first joint committee of the ISO TC97 (Information Processing Systems) and its subcommittees, with the IEC Technical Committee 83 (Information Technology Equipment) and the subcommittee IEC SC47B (Microprocessor systems). The joint committee was formed to eliminate much of the two groups' standardization-activities' overlap and prevent the creation of incompatible standards for the same device or technology area.

Although ISO and IEC are equal partners in the management of JTC 1, most of JTC 1's standards work grew out of ISO's information processing work. In fact, JTC 1 has become one of the most important information technology standards organizations today because so many of the major ISO information technology standards being developed today are actually being produced by JTC 1 groups.

The JTC 1's purpose is to develop international standards in the areas of information technology systems (including microprocessor systems) and equipment. Microprocessor systems include, but are not limited to, microprocessor assemblies, and related hardware and software for controlling the flow of signals at the terminals of microprocessor assemblies.

The JTC 1 initially organized its standards work into four major groupings, each of which contains subcommittees that, in turn contain working groups. The four main groupings and their subcommittees are:

JTC 1 Application Elements Group

SC1: Vocabulary

SC7: Software Engineering

- SC14: Representation of Data Elements
- SC22: Languages

JTC 1 Equipment and Media Group

- SC11: Flexible Magnetic Media for Digital Data Interchange
- SC15: Labeling and File Structure
- SC17: Identification and Credit Cards
- SC23: Optical Disk Cartridges for Information Interchange
- SC28: Office Equipment

JTC 1 Systems Group

- SC6: Telecommunications and Information Exchange Between Systems
- SC13: Interconnection of Equipment
- SC18: Text and Office Systems
- SC21: Information Retrieval, Transfer, and Management for OSI

JTC 1 Systems Support Group

- SC2: Character Sets and Information Coding
- SC24: Computer Graphics
- SC25: Interconnection of Information Technology Equipment (formerly IEC TC83)
- SC26: Microprocessor Systems (formerly IEC TC47B)
- SC27: Security Techniques (grew out of JTC 1 SC20: Data Cryptographic Techniques)

POSIX standardization work is being done within SC22's Working Group 15 (SC22/WG15). A JTC 1 Special Working Group on Strategic Planning is performing a technical study on Application Portability (AP). This study's goal is to identify the standards that need to be written or revised to support application portability between hardware and software environments.

The JTC 1 is not involved in application-specific information technology areas, such as banking and industrial automation systems, nor is it concerned with microprocessor subsystems covered by the scopes of IEC TC22 on power electronics or TC86 on fiber optics.

The JTC 1 has liaison relationships with numerous ISO and IEC Technical Committees, as well as with the CCITT.

Like ISO, membership in JTC 1 consists of delegations from standards organizations in member countries. At present, 23 countries participate in JTC 1, and there are another 11 observer countries. The ANSI holds the secretariat for JTC 1.

For further information, contact: American National Standards Institute (ANSI), 1430 Broadway, New York, NY 10018, (212) 354-3300, Telex: 42 42 96 ANSI UI, or International Organization for Standardization (ISO), Central Secretariat, 1, rue de Varembé, CH-1211, Geneva, Switzerland-40.

SGFS (Special Group on Functional Standardization)

The Special Group on Functional Standardization (SGFS) is an ISO group, under JTC 1, which is responsible for the international standardization process of profiles or functional standards.

C.2.2 Nongovernment Formal Standards Organizations

ECMA : European Computer Manufacturers Association

Established in 1961 to develop data processing standards, ECMA is a trade organization, open to any computer firm developing, manufacturing, or selling in Europe. The ECMA has about 20 members, and approximately 13 active Technical Committees.

ECMA contributes to the ISO standards development efforts, in addition to issuing its own standards. ECMA is particularly active in the development of higher layer protocols for OSI networking. It also is developing a standard for a Portable Common Tool Environment (PCTE).

For further information, contact European Computer Manufacturers Association, 114 rue du Rhone, CH-1204 Geneva, Switzerland, Telephone: 41-22-735-36-34, Telex: 41 3237, Fax: 41 22 786 53 31.

EIA : Electronic Industries Association

The EIA is a US trade organization, whose membership consists primarily of manufacturers. The EIA has been a standards developer in the areas of electrical and electronic products and components since 1926. Many of its standards have been submitted to ANSI and approved as ANSI standards. The EIA is best known for the RS-232-C standard.

For further information, contact John Kinn, Vice President – Engineering, Electronic Industries Association (EIA), 2001 I Street NW, Washington, DC 20036, (202) 467-4961.

IEEE : Institute of Electrical and Electronic Engineers

The IEEE is a professional scientific, engineering, and educational society that develops and publishes standards and specifications in a variety of computer and engineering areas. The standards and specifications published are of three types: true standards, recommended practices, and guides.

''Standards'' are specifications with mandatory requirements. Recommended practices are specifications of procedures and positions preferred by the IEEE. Guides are specifications that suggest alternative approaches to good practice, but make no clear-cut recommendations. The IEEE is accredited by ANSI, and can, therefore, submit its standards directly to the ANSI board of Standards Review. All new and revised IEEE standards are submitted to ANSI for review and adoption as ANSI standards.

The IEEE Standards Board authorizes, coordinates, and approves all standards projects, and coordinates cooperation with other standards organizations. Standards are proposed and sponsored by technical committees of the IEEE Societies, standards committees, or Standards Coordinating Committees (SCC), depending on the scope of the work. Either these committees or standards subcommittees manage the actual standards development and balloting. The individual draft standards are specified in working groups inside the subcommittees—one working group per standard (see Figure C-2).

IEEE membership is open to any dues-paying individuals. Standards participants are individuals, not companies or organizations. IEEE membership is required for voting, but not for participating in the development of draft standards.

Approximately 30 000 members are active in standards development. More than 500 IEEE standards exist, and more than 800 standards projects are underway. The IEEE also administers the secretariat or cosecretariat of 17 American National Standards committees.

The most well known IEEE standards are the IEEE 802.3 CSMA/CD and 802.4 token bus LANS, IEEE-488 bus, the National Electrical Safety Code, and the P1003.n POSIX standards. The 802.3 and 802.4 standards are also approved ISO standards. The core POSIX standard (POSIX.1 {2}) has been approved by ISO, and is now an ISO, as well as an IEEE, standard. The POSIX.0 specifications, with which this document is concerned, will be, in IEEE parlance, a ''Guide'' to a POSIX Open System Environment.

For further information, contact the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

NIST: National Institute of Standards and Technology

The National Institute of Standards and Technology (formerly the National Bureau of Standards) was established by an act of the US Congress on March 3, 1901 to advance, and facilitate the application of, US science and technology for public benefit. Toward this end, the Institute for Computer Sciences and Technology (ICST) within the NIST, conducts research and provides technical advisory services to help Federal agencies acquire and apply computer technology.

The NIST is a major driving force behind standards development. Through the Institute for Computer Sciences and Technology, the NIST develops and publishes Federal Information Processing Standards (FIPS) for the United States. Federal agencies to use in their computer equipment procurements. Federal agencies are obligated to use these standards, where applicable.

Federal computer standards also are widely used by the private sector, and often are adopted as ANSI standards. Besides defining standards, the NIST has defined an Application Portability Profile (APP), which comprises a series of

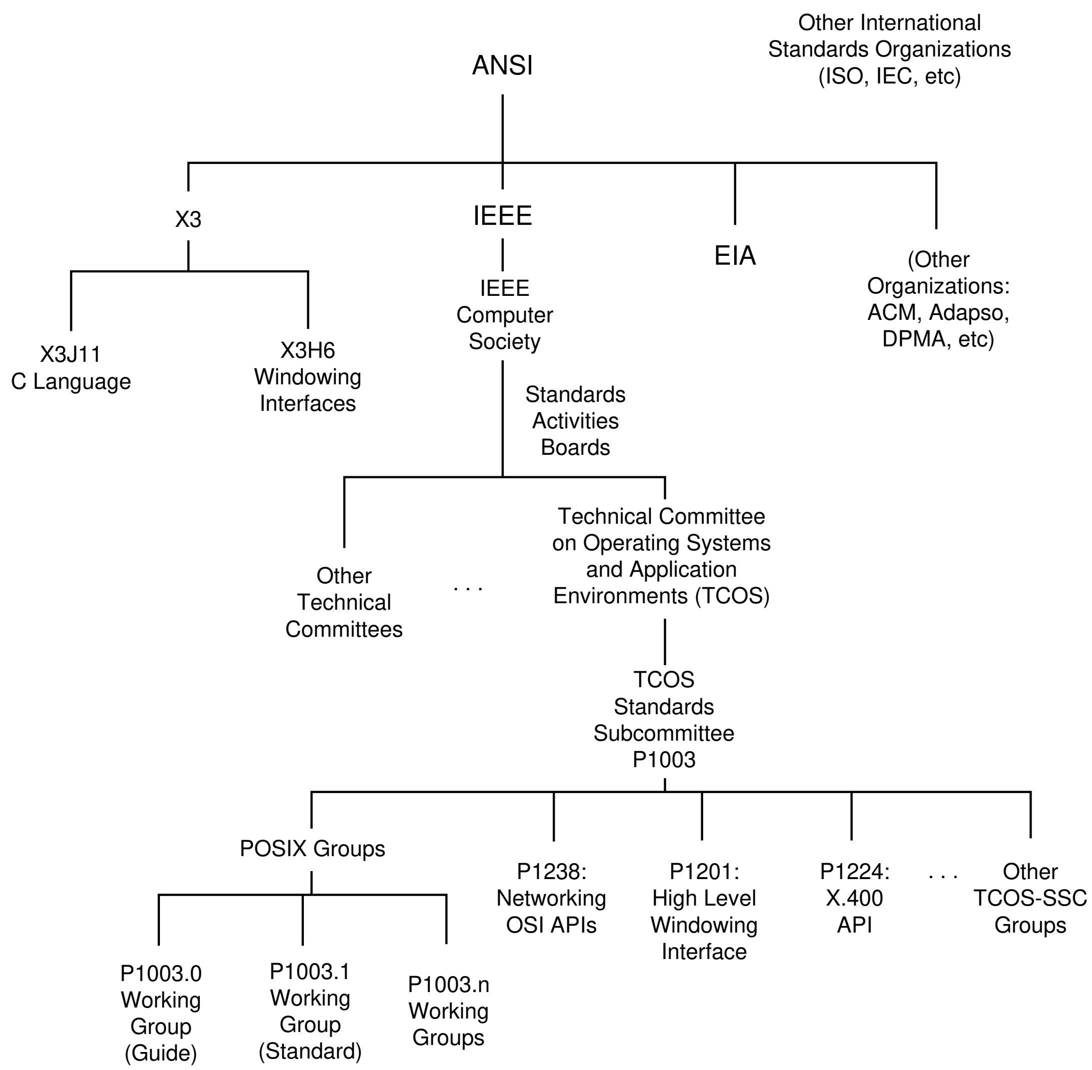

Figure C-2 – IEEE Standards Diagram

nonmandatory specifications and a guide for US government users to use in developing a portable, interoperable architecture and environment.

The development and evolution of both FIPS and the APP is carried out in conjunction with users and vendors through an ongoing series of NIST-conducted Implementor Workshops and User Workshops (e.g., OSI implementors workshops, APP workshops, and Integrated Software Engineering Environment workshops). The workshops provide forums for user and vendor feedback and comments on evolving NIST standards, and help ensure that there is a general commitment among vendors to building products that conform to the evolving NIST specifications.

Additionally, the NIST develops test methods and performance measures to help users and vendors implement standards and to test the conformance of vendor implementations to FIPS specifications. Among others, the NIST has test suites for most FIPS programming languages, FIPS Database SQL, and POSIX.1 {2}. The POSIX.1 {2} conformance test suite, however, is based on the conformance-test assertions developed in the POSIX Test and Methods working group (P1003.3.1).

Besides developing its own standards, NIST staff members participate in a number of other standards activities and organizations, including the ANSI X3 Committee on Information Processing Systems, ISO/IEC JTC 1, CCITT, ECMA, and the IEEE.

For further information, contact the National Institute of Standards and Technology, Gaithersburg, MD 20899, Telephone: (301) 975-2000. E

T1

T1, established in 1984, is an ANSI-accredited standards body that is developing standards and technical reports. The standards and reports are intended to support interconnection and interoperability of telecommunications networks at interfaces with end-user systems, carriers, information and enhanced-service providers, and customer premises equipment.

Six T1 technical subcommittees are currently developing these standards and reports under the T1 Advisory Group. The subcommittees also recommend positions on matters under consideration by other North American and international standards bodies.

T1 Membership and full participation is available to all interested parties. For further information, contact Alvin Lai, Exchange Carriers Standards Association, c/o T1 Secretariat, 5430 Grosvenor Lane, Suite 200, Bethesda, Maryland 20814-2122, or call (301) 654-4505.

X3

X3, established in 1961, is an ANSI-accredited standards body that develops computer, information processing, and office systems standards. X3 also participates in the development of international standards in these areas. In addition, it serves as a Technical Advisory Group (TAG) to ANSI for most of the subcommittees working on international standardization projects within JTC 1. The Computer and Business Equipment Manufacturers Association (CBEMA) functions as X3's secretariat.

X3 membership is open to all organizations, upon payment of a service fee. The current membership includes computer manufacturers, communications carriers, user groups, and government agencies. More than 3200 volunteers from these organizations participate in the X3 standards work. They are organized into about 85 technical groups, working on 700 projects.

Three standing committees report to X3: the Standards Planning and Requirements Committee (SPARC), the Strategic Planning Committee (SPC), and the Secretariat Management Committee (SMC). The following are the major X3 technical committees:

Recognition

X3A1 Optical Character Recognition

Media

X3B5 Digital Magnetic Tape

X3B6 Instrumentation Tape

X3B7 Magnetic Disks

X3B8 Flexible Disk Cartridges

X3B9 Paper/Forms Layout

X3B10 Credit/Identification Cards

X3B11 Optical Digital Data Disks

Data Management and Graphics

X3H2 Database

X3H3 Computer Graphics

X3H3.6 Windowing Interfaces

X3H4 Information Resource & Dictionary

Languages

X3J1 PL/1

X3J2 Basic

X3J3 Fortran

X3J4 COBOL

X3J7 APT

X3J9 Pascal

X3J10 APL

X3J11 C

X3J12 Dibol

X3J13 Common Lisp

X3J14 Forth

X3J15 Databus

Documentation

X3K1 Computer Documentation

X3K5 Vocabulary

Data Representation

X3L2 Codes and Character Sets

X3L5 Labels and file Structure

X3L8 Data Representation

Communication

X3S3 Data Communications

Systems Technology

X3T1 Data Encryption

X3T2 Data Interchange

X3T5 Open Systems Interconnection

X3T9 I/O Interface

Text

X3V1 Office and Publishing Systems

For more information, contact CBEMA, c/o X3 Secretariat, 311 First Street NW, Suite 500, Washington, DC 20001-2178, Telephone: (212) 626-5740.

C.3 Related Organizations

The following organizations are some of the major trade associations, user groups, and professional bodies active in either promoting, implementing, or reviewing information technology standards.

CBEMA: Computer and Business Equipment Manufacturers Association

CBEMA is a trade organization whose primary function is to represent large manufacturers of hardware-based information technologies equipment in lobbying about public policy. In addition, it provides education programs, information exchange forums, and deals with the industry's public image.

CBEMA has long had an interest in standards. It serves as the secretariat for X3. It also offers a standards and technology program where its members can exchange information on standards issues and industry standards.

CBEMA's members are mostly large manufacturers because its dues are tied to corporate revenues and structured in a way that makes it too expensive for small companies to join. Members are either American companies or US subsidiaries of non-American companies.

For more information, contact CBEMA, 311 First Street, NW, Suite 500, Washington, DC 20001-2178, Telephone: (202) 626-5740.

CODASYL : The Conference on Data Systems Languages

The Conference on Data Systems Language (CODASYL) has been active since 1960 in the development of the COBOL language, through its COBOL Committee (CC). Since 1969, it also has been active in the development of a common Data Description Language for defining schemas and subschemas, and in a data manipulation language, through the DBTG Data Base Task Group of the CC. The activities of the CC are documented in the COBOL Journal of Development, which serves as the official COBOL language specification.

In 1969, ANSI (then the United States of America Standards Institute) issued the first COBOL standard. At that time, the X3.4 committee stated that X3.4 recognizes the CODASYL COBOL Committee as the development and maintenance authority for COBOL. In practice, this meant that ANSI agreed not to make any changes to the CODASYL-defined language specification. Although this agreement has been challenged over the years, the CODASYL-ANSI agreement is still strong. As a result, the CODASYL has enormous influence upon the COBOL language.

Toward the end of 1971, a new CODASYL committee was established—the Data Description Language Committee (DDLC). The DDLC was formed to serve the same functions for the schema DDL as the CC does for COBOL. That is, since the schema DDL is a conceptual schema and network-model database language for use with many programming languages, not just COBOL, the DDLC continues the schema DDL development and publishes its own Journal of Development documenting the language's current status.

The COBOL DML and subschema DDL (for defining an external view) of the DBTG are COBOL-specific and have remained part of the CC under the name ''The COBOL Data Base Facility.''

The CODASYL membership is composed of voluntary representatives, mostly from computer manufacturers and users in industry and the US Federal government.

COS : Corporation for Open Systems

COS is a US-based, international, nonprofit association of vendors and users, formed in 1985 to promote and accelerate the adoption of interoperable, multivendor products and services based on OSI and ISDN standards. To accomplish its goals, COS provides a user-vendor forum for the statement of user requirements and the discussion and management of the issues surrounding the deployment of open systems. COS also identifies test requirements, and sponsors test tools development and conformance and interoperability testing to verify that computer products and services conform to OSI or ISDN standards.

COS's membership consists of more than 60 prominent manufacturer, user, and telecommunication service organizations worldwide. COS cooperates with similar organizations such as SPAG Services in Europe and POSI in Japan. Other key groups in the worldwide promotion, implementation and testing of OSI and ISDN standards are affiliated with COS under its Alliance Associate program.

For further information, contact the Corporation for Open Systems, 1750 Old Meadow Road, Suite 400, McLean, VA 22102-4306, USA, Telephone:

(703) 883-2700, Fax: (703) 848-8933. In Europe contact Corporation for Open Systems, Avenue des Arts 1-2, bte 11, 1040 Bruxelles, Belgique, Telephone: 32 2 210 08 11, Fax: 32 2 210 08 00.

E

EPRI : Electric Power Research Institute

The Electric Power Research Institute's (EPRI) is an industry association concerned with electric power utilities. Its membership comprises more than 673 publicly and privately owned utilities in the United States. Besides providing a variety of utility-specific services to its membership, EPRI 's latest mission is to facilitate the use of open systems technology in the utility industry.

Toward this end, EPRI has developed a Utilities Communication Architecture (UCA), which is similar to the National Institute for Standards and Technology's (NIST) Government Open Systems Interconnect Profile (GOSIP) Version 2.0. Much of the UCA was developed by EPRI with the cooperation of Honeywell and Anderson Consulting.

EPRI 's specific open system interests span realtime UNIX , expert systems, and database access using RDA and SQL. Because of the financial structure of the utilities industry, EPRI wants to encourage these and other open systems technologies for equipment with a 12 to 15 year life cycle.

For further information, contact EPRI 's headquarters at 3412 Hillview Avenue, P.O. Box 10412, Palo Alto, CA 94303, Telephone: (415) 934-4212.

ESPRIT (European Strategic Programme for Research and Development in Information Technology)

The European Strategic Programme for Research and development in Information Technology is a European research programme initiative, started in 1982 and sponsored by the Commission of the European Communities. About 227 projects were implemented during the first phase of the project in five major work areas: advanced microelectronics, software engineering and technology, advanced information processing, office automation, and computer integrated manufacturing.

Nearly thirty projects have enabled substantial advances to be made in establishing internationally recognized standards. Examples of the Portable Communications Tool Environment (PCTE) project, the Communication Network for Manufacturing Applications (CNMA) project, and the Herode project, which has prepared an Office Document Architecture standard for adoption as an ISO standard.

The second phase of the Esprit programme will be concerned mainly with three areas—microelectronics and peripheral technologies; the creation of technologies and tools for the design of information processing systems; and enhancing the capacity for using and integrating information technology to extend the scope of its applications.

For further information contact ESPRIT , Director General, DG XIII , CEC , rue de la Loi 200, B-1049 Brussels, Belgium, Telephone: (32 2) 235 11 11, and Telex:

21877 comeu b.

ETSI : European Telecommunications Standards Institute

The European Telecommunications Standards Institute (ETSI), founded in 1988, is a voluntary standards organization involved in producing the telecommunications standards necessary to achieve a European unified market. ETSI was established outside the CEN /CENELEC framework. ETSI , however, works with CEN , CENELEC , and the European Broadcasting Union (EBU) in areas of mutual interest.

ETSI 's voting membership consists of postal administrations, along with manufacturers and trade associations, in each of the CEPT countries. Membership is not restricted to official representatives of member countries. The United States and US companies have been granted observer status.

Standards approved by ETSI are voluntary standards known as ETS (European Telecommunications Standards). ETSI also conducts prestandardization studies, develops technical reports and guidelines, holds conferences, workshops, seminars, and conducts interviews. ETSI 's interim standards are designated I-ETS.

For further information, contact the European Telecommunications Standards Institute, B.P. No. 52, F-06561 Valbonne CEDEX , France, Telephone: (33 92) 94 42 00, Telex: 470 040 F, and Fax Number: (33 93) 65 47 16.

EWOS : European Workshop for Open Systems

The EWOS is an ongoing regional workshop, formed in 1987, to provide and coordinate European input to the international standard profiles process. It was formed as the result of an initiative of SPAG , in conjunction with CEN /CENELEC.

EWOS is the focal point in Europe for the study and development of OSI profiles and corresponding conformance test specifications. EWOS documents have only to be submitted to public enquiry by CEN and CENELEC before becoming European norms.

For further information contact European Workshop on Open Systems (EWOS), rue de Brederode 13, B-1000 Brussels, Belgium, Telephone: 32 2 511 74 55.

INTAP (Interoperability Technology Association for Information Processing)

The Interoperability Technology Association for Information Processing, in Japan, is a national agency, funded by MITI. It deals with information technology, and specifically OSI products and advanced projects. INTAP is developing and providing conformance testing tools and services in Japan in cooperation with POSI.

MAP/ TOP User Group: (Manufacturing Automation Protocol and Technical and Office Protocol)

The MAP Task Force was formed in 1980 by engineers from seven General Motors (GM) divisions, to identify a common OSI-based networking standard for plant-floor systems. The Task Force grew to include all GM divisions, many other users, and many vendors. Its specifications are known as Manufacturing Automation Protocol (MAP).

The MAP specifications mostly reference OSI standards, but they also draw on ANSI, IEEE, EIA, CCITT, and various industry standards. Where standards do not exist, as in the case of the manufacturing messaging protocol, the MAP Task Force is either defining its own or instigating their formation by industry standards bodies.

In 1984, the MAP Users Group was formed, under the auspices of GM, with the Society of Manufacturing Engineers as its Secretariat. Its objective is to promote knowledge and use of MAP, and to insure input from users.

In 1985, Boeing sponsored a similar effort to specify common networking protocols, known as the Technical and Office Protocols (TOP), for the engineering and business offices. TOP is largely compatible with MAP, differing only at the lower two layers and the application layer where TOP addresses requirements of the technical and office user, rather than factory users.

Later in 1985, a TOP Users Group was formed. The entire effort became an international effort known as MAP/TOP, and the user group became the MAP/TOP User Group, which meets twice a year.

Today, the MAP/TOP User Group is an independent, self-funded organization that represents thousands of users worldwide, tied together through a worldwide federation of MAP/TOP user groups. Membership is open to either users or companies. The Industry Cooperative Services (ICS) is the worldwide secretariat. The MAP/TOP User Group also is a member of the Corporation for Open Systems (COS) and in North America, COS acts as the MAP/TOP User Group secretariat.

The MAP/TOP User Group is a Requirements Interest Group (RIG) of the Corporation for Open Systems (COS). This means that the MAP/TOP User Group generates requirements that vendors can use to built products. COS serves as the coordinator between users and vendors.

The MAP/TOP Task Force and User Group also is a major contributor of technical and conceptual ideas and specifications to the NIST, COS, and many of the IEEE POSIX Groups.

For further information contact the World Federation of MAP/TOP Users Groups, P.O. Box 1457, Ann Arbor, MI 48106, Telephone: (313) 769-4571, Fax: (313) 769-4064. In North America, also contact the Corporation for Open Systems at 1750 Old Meadow Road, Suite 400, McLean, VA 22102-4306, Telephone: (703) 883-2700, Fax: (703) 848-8933.

E

Network Management Forum

A vendor-driven group, the Network Management Forum is chartered to produce a commonly agreed-upon specification subset of ISO's network management protocols and the command sets to implement this subset. The promise of the NMF is that all of the network management products that its members come up with will conform to this common specification.

The NMF itself will produce no products nor will it specify implementations. Rather, the NMF will specify interfaces.

Major vendors belong to the NMF from both the computer and telecommunications industries. The NMF has published Release 1 of its specifications (1990). Member firms are developing products that conform to Release 1.

NMF information may be had from the organization at 40 Morristown Road, Bernardsville, NJ 07924. Telephone: (201) 766-1544.

NPSC: National Protocol Support Center

An Australian organization, the National Protocol Support Center was formed in 1986 as a joint effort between industry and the government. Like SPAG, COS, and POSI, the NPSC is promoting the adoption of OSI standards in information technology products and will be supporting a conformance testing capability in Australia. The Australian government, however, provides approximately 50 percent of the NPSC funding. For further information, contact (contact address and other information TBD).

Object Management Group

Founded in 1989 and headquartered in Framingham, MA, with marketing operations in Boulder, CO, the Object Management Group (OMG) is an international organization of more than 146 systems vendors, software developers and users. The OMG was founded to promote the theory and practice of object management technology in the development of software.

The OMG's goal is to develop a common framework, based on industry-derived guidelines, that is suitable for object-oriented applications. The adoption of this framework will make it possible to develop a heterogeneous applications environment across all major hardware and operating systems.

The OMG members are quick to form a consensus on certain object management issues because they see these issues directly affecting their software sales. For example, the OMG's object request broker design—key software needed to allow disparate open systems to request object services from remote sites—is supported by most major object-oriented software vendors.

Further information is available from the OMG at 492 Old Connecticut Path, Framingham, MA 01701. Telephone: (508) 820-4300.

OSF : Open Software Foundation

The Open Software Foundation is a nonprofit, international consortium. Its goals include the development of software specifications and test suites for an open computing environment.

OSF specifications are defined, and software developed, using an open process into which vendors and users have input and access. The resulting AES specifications will be available in the public domain, and the software licensable, to OSF members and nonmembers, under identical terms. Both members and non-members can also submit technologies to the OSF for consideration as an OSF specification and/or offering. OSF's specifications and software will be based on the ISO/IEC 9945-1 core POSIX standard (POSIX.1 {2}), a variety of international, national, and industry standards and other consortia specifications. The remainder of OSF software and specifications will be based on technologies contributed by numerous companies and universities as part of OSF's Request for Technology (RFT) process.

OSF active-participation membership is open to user organizations, computer hardware and software suppliers, government agencies, educational institutions, and other interested organizations worldwide. For further information, contact OSF at Eleven Cambridge Center, Cambridge, MA, Telephone: (617) 621-8700. Alternatively, contact European headquarters at Open Software Foundation/Europe, Stefan-George-Ring 29, 8000 Munich 81, Germany, Telephone: (49 89) 930 920, or Open Software Foundation/Japan, ABS Building, 2-4-16 Kudan Minami, Chiyoda-Ku, Tokyo 102, Japan, (81 3) 3 221 9770.

Petrotechnical Open Software Corporation

Founded in October, 1990, the Petrotechnical Open Software Corporation (POSC) was started by BP Exploration, Chevron, Elf Aquitane, Mobil and Texaco to facilitate the development of integrated computing technology for the exploration and production (E & P) segment of the international petroleum industry. Today, membership is open to all entities interested in the E & P industry. These members include other petroleum companies, E & P service companies, software vendors, computer manufacturers, and research institutes.

POSC's primary mission is the development of an industry-standard, open systems-based, software integration profile for E & P applications. This platform will be the interface between petrochemical software applications, database management systems, workstations and users. POSC activities focus on the development of an integrated E & P data model, a common look and feel user front-end, and a set of test suites enabling developers to evaluate their offerings against selected industry standards.

POSC is moving quickly and has sent out two public requests for inputs in several technical areas. Project teams for base standards, the E & P data model, and data access are in place. Staffing is in progress for other projects and special interest groups have been formed. POSC offerings will be released to industry for production over the next few years.

POSC is headquartered in Houston, TX at 10777 Westheimer, Suite 275, Houston, 77042. Telephone: (713) 784-1880.

POSI : Promoting Conference for OSI

The Promoting Conference for OSI was formed in Japan in November 1985 by six major computer manufacturers and the Nippon Telephone and Telegraph Corporation. Its raison d'etre is to promote the adoption of OSI standards by cooperating with other international groups that have the same objective, such as the European-based SPAG and the US-based COS. But conformance testing in Japan is being developed and will be provided by the INTAP.

For further information, contact (contact information TBD).

SPAG : Standards Promotion and Application Group

The Standards Promotion and Application Group (SPAG), founded in 1983, is a nonprofit, international research and development consortium of about 65 information technology manufacturers and users. In 1986, it became a company registered under Belgian law as SPAG Services s.a. SPAG's goals are to promote multivendor, interoperable products based on international standards, particularly OSI, and to keep its members informed about the latest developments in functional standards and conformance testing of products.

To achieve its goals, SPAG plays a leading role in the European Workshop on Open Systems (EWOS), publishes the Guide to the Use of Standards (GUS) regularly, and participates in the development of International Standard Profiles (ISPs). SPAG is particularly active in the development and marketing of test tools for manufacturing applications. Through its SPAG-CCT efforts, (a collaboration of European organizations) which arose out of the ESPRIT Project 955, SPAG is developing, and will be providing, conformance test tools for testing MAP/TOP 3.0, and conformance testing services to industry.

SPAG also is working within Europe to implement the certification infrastructure for OSI products, and is involved in a number of Conformance Test Services (CTS) projects within the Commission of European Communities (CEC). In addition, SPAG is active in Telecommunications areas and is leading a consortium developing verification services for the Broadband Networks project RACE.

Twelve shareholder companies make up SPAG's board of directors. The original founding companies—Bull, ICL, Nixdorf, Olivetti, Philips, Siemens, and STET—occupy seven seats on SPAG's twelve member board. The shareholder membership was subsequently expanded to include Alcatel, British Telecom, Digital Equipment Corp., Hewlett-Packard, and IBM, who occupy the five remaining board seats.

SPAG has close working relationships with its counterparts in North America (COS) and the Far East (POSI).

For further information, contact Graham Knight, at SPAG Services s.a., Standards Promotion and Application Group (SPAG), Avenue des Arts, 1-2 bte 11, 1040 Brussels, Belgium, Telephone: 32 2 210 08 11, Fax 32 2 210 08 00.

SQL Access Group

The SQL Access Group is a vendor group formed by a number of people in the ISO Remote Data Access (RDA) Group.

The SQL Access Group's charter is several fold. First, the Group is chartered to define a common subset of SQL functions to reconcile the many SQLs that exist. The specifications will include an SQL data format, as well as protocols for moving data within a multivendor SQL environment. Also included will be specifications for an enhanced SQL programming interface that will let developers write a single application that can access a variety of SQL databases. These SQL Access specifications are scheduled to be published in late 1991.

The SQL Access Group's second charter is to accelerate the work of the RDA group. Third, the SQL Access Group is working on putting more distributed functionality into RDA. Toward this end, each thing accomplished by the SQL Access group is fed back into the RDA group.

For further information, contact the SQL Access Group at (Address TBD).

UniForum

UniForum is a nonprofit international association of open systems professionals. Founded in 1980 as /usr/group, the association has, through its standards committees and technical committees, provided contributions to various standards and continues to be involved in the formal standards development process. The specifications and standards to which UniForum has contributed include:

— The 1984 /usr/group Standard was contributed as a base document for the IEEE P1003.1 work.

— The UniForum Technical Committee on Real Time meets jointly with the IEEE P1003.4 working group, working on the emerging POSIX realtime standards.

— The UniForum Technical Committee on Supercomputing evolved into the IEEE P1003.10 working group.

— The UniForum Technical Committee on Transaction Processing evolved into the IEEE P1003.11 working group.

— The UniForum Technical Committee on Internationalization has contributed specifications to the IEEE P1003.1 and P1003.2 working groups and the ANSI X3J11 standard C committee and continues to be a technical resource for both formal and informal standards development bodies.

UNIX International/ UNIX System Laboratories

UNIX International (UI) is a nonprofit industry organization formed to represent hardware manufacturers, system integrators, independent software vendors, value-added resellers, end-users, government agencies worldwide, industry standards bodies, and academic and research institutions who want to direct the evolution of System V UNIX and its corresponding specification, the *System V*

Interface Definition (SVID). It has since expanded its scope to provide a framework for UNIX-based open systems work in the areas of desktop computing, corporate hub computing, distributed computing, and an enterprise-wide framework known as ''Atlas.''

Unlike X/Open, OSF, AT&T, and the IEEE, UI does not produce specifications, software, or standards. Its functions range from specifying technical and timing requirements for future System V versions and making suggestions about specific function designs to influencing AT&T UNIX licensing policies.

Using its ''one-member, one-vote'' approach, UI members formulate a consensus regarding the requirements and technical specifications for new System V UNIX versions. UI delivers its requirements to UNIX System Laboratories (USL), the AT&T spinoff that develops, distributed, and licenses UNIX. UI is USL's primary input source on technical requirements, conformance, and timing of releases. USL is committed to implement software to satisfy UI's requirements, unless there is a reason not to.

For further information, contact UNIX International, Waterview Boulevard, Parsippany, NJ 07054, (201) 263-8400 or (800) 848-6495. In Europe, contact UNIX International, Avenue de Beautieu 25, 1160 Brussels, Belgium, (32-2-672-3700). In the Asian Pacific region, contact Karufuru-Kanda Bldg. 8F, 1-2-1 Kanda Suda-cho, Chiyoda-du Tokyo 101, Japan, (81) 3-5256-6959.

User Alliance for Open Systems

The User Alliance for Open Systems was formed from two informal organizations (the Atlanta 17 and the Houston 30). The Alliance is currently a Requirements Interest Group (RIG) of the Corporation for Open Systems International (COS).

The Alliance is dedicated to overcoming barriers to open systems and speeding the development and deployment of open systems products. It intends to act as a catalyst toward the development and use of open systems. It will develop no specifications or products. Rather, the Alliance will create and support processes to influence and accelerate the availability of open systems technology (e.g., a repository of information about the cost benefits of open systems).

In 1990 the organization began its work by identifying barriers to open systems and global actions to eliminate those barriers. In 1991 the organization intends to start bringing resources to bear to achieve its goals. The Alliance has had one formal meeting (Dallas, March 1991) and will have its second formal meeting in McLean, Virginia in Nov. 1992. Alliance committee work is ongoing throughout this period with three major subgroups in the formative stages.

For further information, contact the Corporation for Open Systems, 1750 Old Meadow Road, Suite 400, McLean, VA 22102-4306, Telephone: (703) 883-2700.

X.400 API Association

The X.400 API (Application Programming Interface) Association is an industry association formed initially to bring X.400 messaging into the PC LAN world. There are more than twenty companies in the association, and they include most of the current X.400 players.

Among its activities, the X.400 API Association developed an X.400 Application Programming Interface specification in conjunction with X/Open. These interfaces, completed in September 1990, are jointly owned by the X.400 API Association and X/Open. The two organizations contributed these interface specifications to the P1224 Group to use as a basis for the P1224 standard.

For further information contact (Address and other contact information: TBD)

X/Open

X/Open is an independent, nonprofit consortium formed in 1984. Its goals are to determine user and market requirements and to specify a complete, source-level-portable application environment and test suites. Although its members were initially vendors, X/Open's membership now encompasses users, system integrators, value-added resellers, government agencies worldwide, other industry-standards groups, and academic and research institutions.

The X/Open environment includes specifications for an operating system interface, networking, data management, programming languages, floppy disk formats, internationalization, and distributed transaction processing. The X/Open Group does not normally define standards for these areas. Instead, it chooses from existing and emerging standards. An X/Open market research program and open user requirements congress identify and prioritize user and market requirements, based on input solicited from users. These prioritized requirements are published in a document known as the *Open Systems Directive*. These prioritized requirements also help drive the X/Open specification process. The X/Open specifications are published in a series of books known as the X/Open Portability Guide.

The X/Open environment is based on the ISO/IEC 9945-1 core POSIX (POSIX.1 {2}) standard, parts of AT&T's System V Interface Definition (SVID), and formal international standards that are already accepted or likely to be accepted. However, to rapidly get standards into the field for practical use, where no formal standards exist, X/Open specifies industry standards and widely-accepted de facto standards (including some based on real-world products that have achieved consensus in the marketplace). In some instances where neither formal nor de facto specifications exist but there is a strong need for standards (e.g., internationalization and transaction processing), X/Open has itself defined specifications.

For further information, contact X/Open Company Ltd. at Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, UK, Telephone: 44 734 508 311. In the US, contact X/Open at 1010 El Camino Real, Menlo Park, CA 94025, Telephone:

(415) 323-7992.

Annex D
(informative)

Electronic-Mail

Responsibility: Kevin Lewis

The following table lists currently-known e-mail addresses for active working group members. To correct your entry, send e-mail directly to Hal Jespersen, listed below.

Michelle Aden	Sun Microsystems	`aden@ebay.sun.com`	
Carolyn Baker	MITRE	`cgb@d74sun.mitre.org`	
Timothy Baker	Ford Aerospace		
Ralph Barker	UniForum	`ralph@techcomm.uniforum.org`	E
Rich Bergman	NOSC	`rich@tecr.nosc.mil`	
Andy Bihain	GTE Telops	`arb1@bunny.gte.com`	E
Jacques Cazier	Mitre	`cazier@mitre.org`	E
Bud Conrad	Tandem	`conrad_bud@tandem.com`	E
Joseph Cote	Treasury Board of Canada	`tbsitm@nrcvm01`	E
Bernard Cox	NASA JSC		
Francis Deckelman	US Navy	`deckelman@a.151.edo`	E
Matt Einseln	Datafocus		E
Don Folland	CCTA	`def@cctal.co.uk`	
David Folsom	CDC	`dbf@udlv.cdc.com`	E
Thomas Ford	USAF	`tford@xpt.ssc.af.mil`	
Bob Gambrel	Unisys	`rjg@rsvl.unisys.com`	
Al Hankinson	NIST /NCSL	`alhank@swe.ncsl.nist.gov`	
E. Lee Hutchins	USAF	`thutch@ssmct62.ssc.af.mil`	E
Jim Isaak	DEC	`isaak@decvax.dec.com`	
Petr Janecek	X/Open	`p.janecek@xopen.co.uk`	
Astrid Jeffries	Unisys	`astridj@convergent.com`	E
Hal Jespersen	POSIX Software Group	`hlj@posix.com`	
Lorraine Kevra	AT&T	`L.Kevra@att.com`	
Ruth Klein	AT&T	`ruthlk@attunix.att.com`	
Doris Lebovits	AT&T	`lebovits@attunix.att.com`	
Kevin Leininger	Fermilab	`kevin@fnalf.fnal.gov`	E
Kevin Lewis	DEC	`klewis@gucci.dec.com`	

Heinz Lycklama	Interactive Systems	heinz@ism.isc.com	
Randolph Lynwood	NASA		
Doug MacDonald	General Electric		
Roger Martin	NIST	rmartin@swe.ncsl.nist.gov	
Dick McNaney	SAIC	saic-02@huachuca-emh2.army.mil	
Pete Meier	IBM	...uunet!aixsm!meier	
Howard Michel	USAF	michelhe@hqafsc-vax.af.mil	E
Gary Miller	IBM	...uunet!aixsm!miller	E
Kevin Murphy	BT	murphy_k_v@bt-web.bt.co.uk	E
Mary Lynne Nielsen	IEEE	m.nielsen@ieee.org	
Patricia Oberndorf	NADC	tricia@nadc.navy.mil	
Jim Oblinger	NUSC	oblinger@ada.nusc.navy.mil	E
Pat Patterson	NASA	patterso@gmuvax2.gmu.edu	
David Pruett	NASA JSC	dpruett@nasamail.nasa.gov	
Wendy Rauch	Emerging Technologies Group	...uunet!etg!wrauch	
Lynwood Randolph	NASA HQ	randolph@nasamail.nasa.gov	E
Brad Reed	EDS	reed@eds.com	
Gregory Sawyer	Space & Naval Warfare Systems Command		E
Carl Schmiedekamp	NADC	schmiede@nadc.navy.mil	
Fritz Schulz	OSF	fschulz@osf.org	
Richard Scott	Chemical Abstracts Service	uunet!osu-cis!chemabs!rls27	
Glen Seeds	Systemhouse		
Charles Severance	Mich. State Univ.	crs@convex.cl.msu.edu	
Lewis Shannon	NCR	lew.shannon@dayton.ncr.com	
Peter Smith	DEC	psmith@decvax.dec.com	
Keith Stobie	Tandem	stobie_keith@tandem.com	E
Sandra Swearingen	USAF	tic-tisc@afcc-oal.af.mil	
Jong Sung Sunwoo	NCA		E
Marti Szczur	NASA/GSFC	msxcxur@postman.gsfc.nasa.gov	
Ravi Tavakley	CDC	ravi@kiran.under.cdc.com	E
Martial Van Neste	CGI Group	vanneste@bond.crim.ca	
Robert Voigt	Space & Naval Warfare Systems Command	voigt@nusc.ada.arpa	
Gentry Watson	UNIX Int'l	glw@ui.org	
Alan Weaver	IBM	...uunet!aixsm!weaver	
John Wilbur			E
John Williams	GM-CPC Hdqts.	jwill08@c4.eds.com	E
Arnold Winkler	Unisys	winkler@pre.unisys.com	E
George Zerdian	Hughes	george@eos.wel.scg.hac.com	

Annex E
(informative)
Additional Material

E.1 Software Development Environments

Responsibility: Don Folland

E.1.1 Overview and Rationale

Software Development Environments are dealt with as a particular application area needing special attention for the following reasons:

- The domain of Software Development Environments is one of prime importance. Software development is a major area of expenditure for government and large commercial organizations.
- The need for standardization is being driven not only by the SDE vendors and users, but also by the Independent tool developers who want to get their tool products on as many vendor platforms as possible.
- The SDE domain calls not only for portability, but also for particular integration and interoperability requirements.
- The domain is primarily of interest to that user community that has large complex software development requirements, but it is also of interest to all application areas as software development is an enabling technology for all applications.

Software engineers seek more powerful assistance to improve productivity and quality in the software development process. Considered opinion currently favors Project Support Environments (PSE) underpinned in such a way that the facilities are capable of being implemented on different machines. A PSE needs a base holding information such as specifications, designs, code, schedules, configuration plans, tests, etc., to support the developers. The interface between the base and the tools must ensure portability of the tools. Again, these tools will be supported by relevant language standards.

Certain design methodologies themselves have been modeled formally to establish their degree of rigor and self-consistency. Function Point Analysis is one method of measuring software systems and computing productivity that is increasing in use. It measures inputs, outputs, and entities accessed to determine transaction size; it gauges technical complexity by reference to 19 characteristics. These are combined to give a measure of systems size. Productivity is the ratio of system size in function points to the effort required to produce or maintain the system.

Generally, software support for the development process is in its infancy and effective metrics have not yet been developed.

E.1.2 Scope

The problem domain is complex software development, from the generation of an idea to the delivery and ongoing support of a solution product set.

Thus, an SDE may include some or all of the following:

(1) Software Development Life Cycle
   (a) Requirements analysis
   (b) Logical design
   (c) Physical design
   (d) Functional and interface specification
(2) Activity support
   (a) Prototyping
   (b) Program development and testing
   (c) Quality assurance and regression testing
   (d) Generation of user documentation
   (e) User training
   (f) Problem report tracking and maintenance
   (g) Maintenance and tracking of schedules
(3) Configuration Management
   (a) Automatic version management
   (b) Integrity management
   (c) Traceability
(4) Project Management
(5) Data Administration
   (a) Access control

In the context of developing software for a POSIX Open System Environment, design will take account of portability and interoperability requirements. The SDE tools themselves should be portable. The software development activities may be provided with a large set of tools and applications. The SDE must provide the necessary support for the integration of all of these tools.

E.1.3 Reference Model

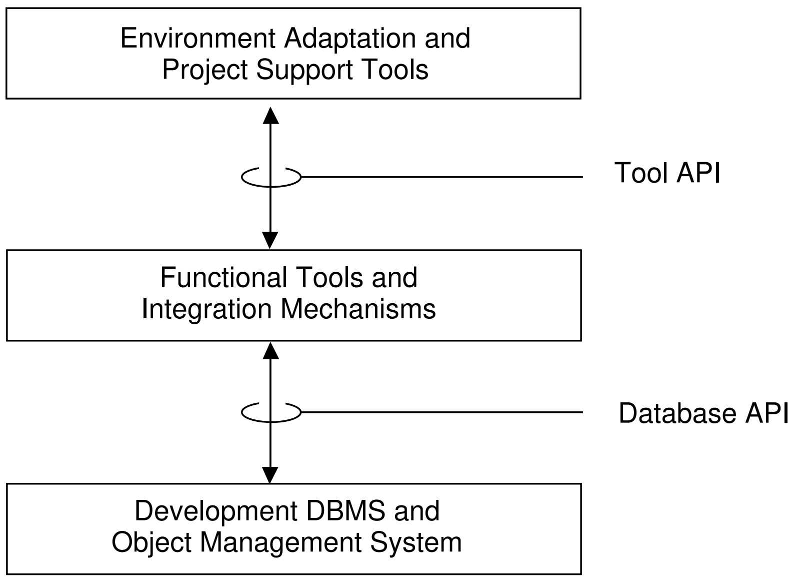

Figure E-1 – Software Development Model

In this clause the conceptual view of software development is related to the POSIX Reference Model (Figure 3-1). The software developer's view is shown in Figure E-1. The tools used to develop software can be viewed as applications in their own right in the context of the POSIX Reference Model, requiring the same services from the platform as for Database Management.

In the Software Development Model, the Environment Adaptation and Project Support Tools ''layer'' provides the essential link between the programmer, designer or analyst, the design method, and the development infrastructure. At this level are provided the tools and applications that are unique to the project or methodology; e.g., project management workbench. It requires support from a consistent human-computer interface to the Functional Tools.

The Functional Tools and Integration Mechanisms embrace the essential tool set to enable software developers to build software. It includes simple tools such as editors, tools for tool-building, and integration mechanisms. There will be tools for Configuration Management, Version Management, and System Administration. It is not within the scope of this guide to discuss these in detail.

The whole software development environment is underpinned by essential management systems, such as object management system, a data dictionary, a user interface management system, and environment management. A database will frequently be established to hold specifications, designs, configuration plans, etc.

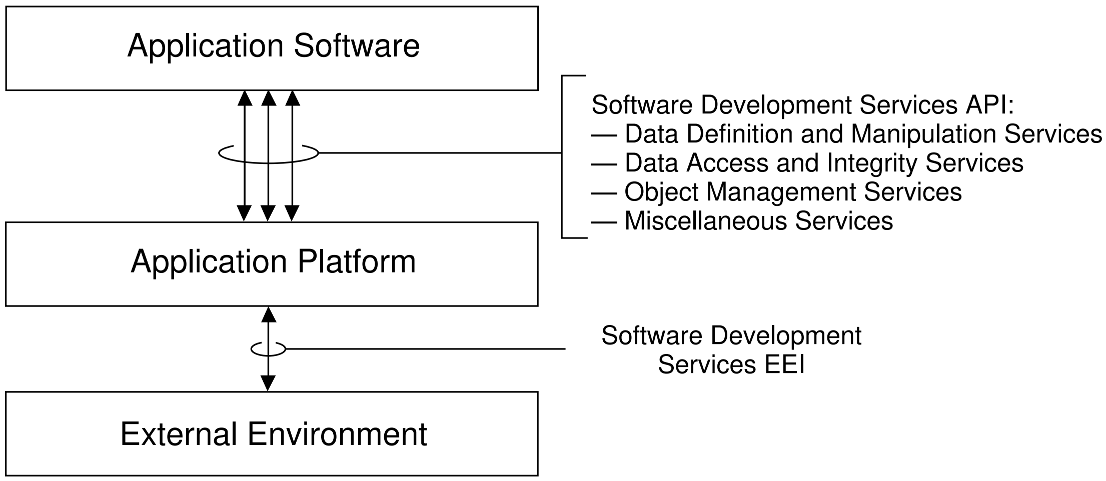

Figure E-2 – Software Development Reference Model

In the POSIX Open System Environment, the software development model can be incorporated into the POSIX Reference Model as in Figure E-2. The model shows that the tools and services required by the software developer are part of the POSIX Open System Environment and are available through the POSIX OSE API.

E.1.4 Services Requirements

Software developers, i.e., designers, analysts, and programmers, use software applications to facilitate the complex task of software development. A tool will require services from the application platform and will frequently require support from another application in the application set. There are many possible implementations of tool sets. Descriptions of these are beyond the scope of this guide.

E.1.4.1 Application Program Interface Services

The services required at the API are essentially similar to those described for Database Management in 4.4.4.1; i.e., Data Definition and Manipulation, Data Access, Data Integrity, and such Miscellaneous Services as Data Dictionary.

E.1.4.2 External Environment Interface Services

A consistent human-computer interface to the tool set is required. Some of the programmer's tool set will be explicitly focused on windowing services (such as 4.7 and 4.8) and provide assistance to develop software with improved usability.

Resource data formats must be specified in order to ensure effective information interchange [for example, CASE Data Interchange Format (CDIF)], for which standards are currently under development under the aegis of the CDIF Technical

Committee (see also E.1.5.2 and 4.5).

Protocol services are required for the transport of data (see 4.3).

E.1.4.3 Interapplication Software Entity Services

Many of the tools depend for interface between one another upon the data dictionary/repository, which is a key software component and which may conceptually be regarded as part of the Applications Platform. Included in this category will be utilities for servicing the DBMS, such as recovery, reorganization, and restructure:

— Object management system

— User interface management system

— Database management system

— Transaction processing management system

Details of these management systems may be recorded in the data dictionary/repository.

E.1.4.4 Software Development Resource Management Services

These services are generally not visible to the programmer or software developer at the Tools API, usually being provided by the tool building and other software development utilities.

E.1.5 Standards, Specifications, and Gaps

This subclause describes current accepted standards that are relevant to this area in addition to the language standards in 4.1.5 and the database standards in 4.4.5.

E.1.5.1 Current Standards

This subclause briefly identifies the current standards in this area.

The following provides place holders for further text to be inserted – assistance required please.

E.1.5.1.1 International Standards

Labeling and File Structure of Magnetic Media

The following standards refer to the labeling of magnetic media and for the file structure on such media to facilitate information interchange:

Labeling of magnetic tape	ISO 1001
Labeling of cassette and cartridge	ISO 4341

Table E-1 – Software Development Standards

Service	Specification	Subclause
Miscellaneous Services:		
Labeling of magnetic tape	ISO 1001	4.11.5.?
Labeling of cassette and cartridge	ISO 4341	4.11.5.?
Labeling of flexible disks	ISO 7665	4.11.5.?
Volume and file structure for flexible disks	ISO 9293	4.11.5.?
Volume and file structure for CD-ROM	ISO 9660	4.11.5.?
Documentation symbols and flowchart conventions	ISO 5807	4.11.5.?
Documentation of applications	ISO 6592	4.11.5.?
Program flow for sequential files	ISO 6593	4.11.5.?
Data descriptive file for information interchange	ISO 8211	4.11.5.?
Program constructs and conventions	ISO 8631	4.11.5.?
User documentation	ISO 9127	4.11.5.?

Labeling of flexible disks	ISO 7665
Volume and file structure for flexible disks	ISO 9293
Volume and file structure for CD-ROM	ISO 9660
Data descriptive file for information interchange	ISO 8211

The above-mentioned standards might be more suitably called out in Richard Scott's section 4.5.

Software Documentation

There are several standards dealing with documentation to assist with the task of software development, and therefore potentially facilitating programmer and designer portability, as well as user documentation.

Documentation symbols and conventions for data, program and system flowcharts, program network charts, and system resources charts	ISO 5807
Guidelines for the documentation of computer-based application systems	ISO 6592
Program flow for processing sequential files in terms of record groups	ISO 6593
Program constructs and conventions for their representation	ISO 8631
User documentation and cover information for consumer software packages	ISO 9127

E.1.5.1.2 Regional Standards

ECMA has approved ECMA-149 as the standard for the Portable Common Tool Environment (PCTE).

E.1.5.1.3 National Standards

To Be Provided

E.1.5.2 Emerging Standards

This subclause describes the activities currently in progress to further standardize this area.

E.1.5.2.1 International Standards

To Be Provided

E.1.5.2.2 Regional Standards

To Be Provided

CASE Data Interchange Format (CDIF)

The CDIF Technical Committee is developing a data interchange format to serve as an industry standard for exchanging information between Computer-Aided Software Engineering (CASE) tools. CDIF is an EIA-endorsed initiative. It assumes that two or more tools may interface asynchronously with each other and will transfer information from one to another via ''CDIF files.'' The types of information that may be contained in these files is defined by the CDIF Conceptual Models.

Portable Common Tool Environment (PCTE)

ECMA TC33 has responsibility for the development and maintenance of PCTE. The committee formed a Task Group in 1988 to develop a Reference Model which would assist the standardization process. Such a model has been developed totally independent of PCTE, and is described in ECMA Technical Report 55. The model provides a way to describe, compare, and contrast CASE environment frameworks.

E.1.5.2.3 National Standards

To Be Provided

E.1.5.2.4 National Standards

To Be Provided

E.1.5.3 Gaps in Available Standards

E.1.5.3.1 Public Specifications

To Be Provided

E.1.5.3.2 Unsatisfied Service Requirements

To Be Provided

E.1.6 OSE Cross-Category Services

Not applicable.

E.1.7 Related Standards

To Be Provided

E.1.8 Open Issues

— Relationship between methodology and formats

[PCTE and CAIS-A have been moved here largely because it is not clear what to do with them. They are not adequately accommodated by this model. They are both hybrids of operating system and database management system capabilities that seem to belong either everywhere or nowhere. They could both well be used in conjunction with a P1003.1 implementation, but they could also be implemented on other base operating systems, or implementations could even expand their capabilities to provide full operating systems. P1003.0 must decide what to do with them.]

PCTE

An effort by the European Computer Manufacturers Association (ECMA) has resulted in the definition by Technical Committee 33 of the Basis for the Portable Common Tools Environment (PCTE). This is now an ECMA standard and is referred to as Standard ECMA-149.

CAIS-A

MIL-STD-1838A (CAIS-A) was developed by the US Department of Defense to provide a common foundation for Ada Programming Support Environments. Similar in nature to PCTE (see above), it too covers many of the system services covered by 4.2.4. In addition, it provides data management services such as those discussed in 4.4 and data interchange services (specifically, a Common External Form) similar to those discussed in 4.5.

Alphabetic Topical Index

A

B

C

D

E

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Acknowledgments

We wish to thank the following organizations for donating significant computer, printing, and editing resources to the production of this standard: the X/Open Company, Ltd.

Also we wish to thank the organizations employing the members of the Working Group and the Balloting Group for both covering the expenses related to attending and participating in meetings, and donating the time required both in and out of meetings for this effort.

Editor's Note: This list should be the union of the companies sponsoring Working Group attendees and Balloting Group members. It will appear after balloting begins.

<*to be provided* > <*to be provided* >

In the preceding list, the organizations marked with an asterisk (*) have hosted 1003 Working Group meetings since the group's inception in 1985, providing useful logistical support for the ongoing work of the committees.