Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

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]

Guide to the POSIX Open System s 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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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. E E E E

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 Varembe´, Case Postale 56, CH-1211, Genè ve 20, Switzerland/Suisse.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

48 ## 1.4 Test Methods E

49 Not applicable. E

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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* .

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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. E E

E

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.

E

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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 API s 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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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. E E E

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 non-standard.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

(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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

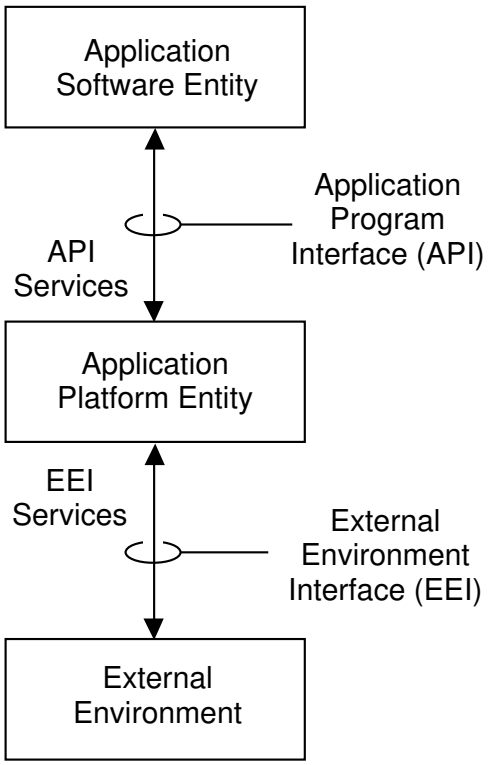

Figure 3-1 – POSIX OSE Reference Model

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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

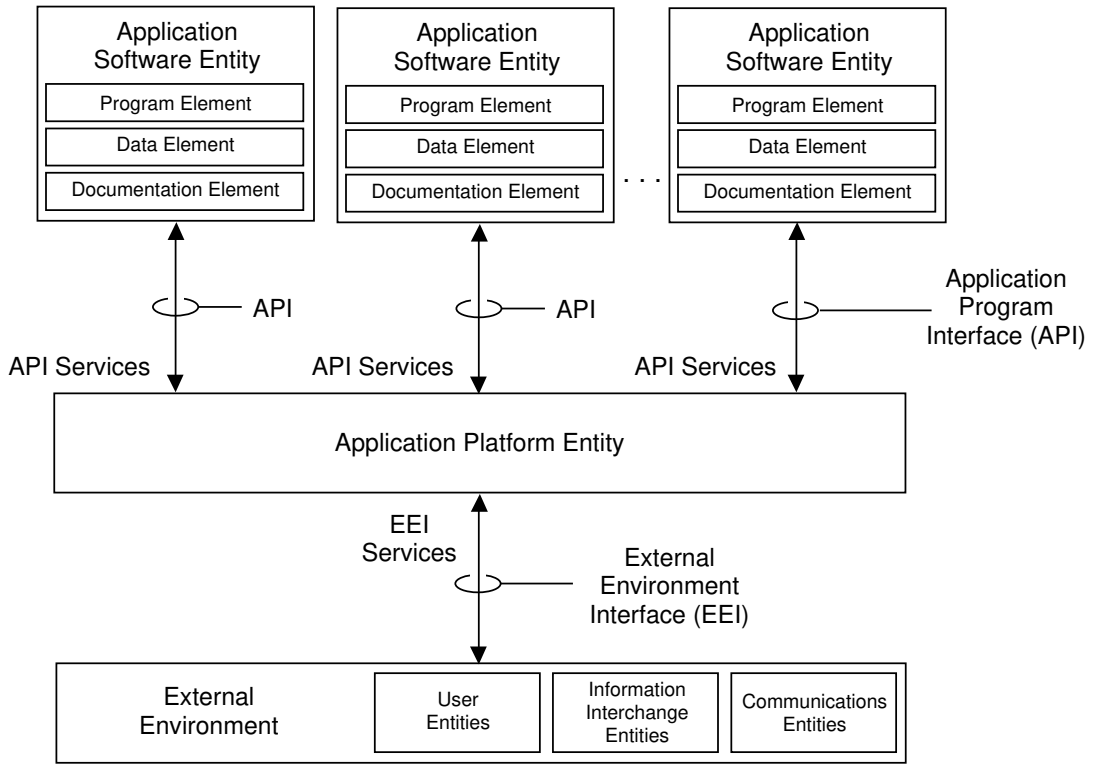

Figure 3-2 – POSIX OSE Reference Model — Entities

— 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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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).

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.