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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

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

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

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 E

PEX — PHIGS Extensions to X E

PEX is a network protocol extension to the X Window System. As many E
applications require 3-D graphics and other forms of input devices such E
as dials and button boxes, all of which are not supported by X, it became E
necessary to extend the X Protocol to include 3-D graphics. PHIGS was E
selected as the application program interface because of its acceptance E
as a 3-D standard, its high degree of input ability, and its powerful E
database editing capabilities. In 1988, the MIT X Consortium contracted E
to add 3-D and extended input extensions to the X protocol and the first E
release of PEX as a sample implementation (PEX-SI) was made in Janu- E
ary 1991 but is not yet available commercially. Using PEX, PHIGS E
workstations would be defined as X Windows. For the programmer, X, E
PHIGS, and PEX standards provide portability. E

4.8.5.3.2 Unsatisfied Service Requirements E

— 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.

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

— 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 DEC net 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.

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

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

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

4.9 Character-Based User Interface Services

Responsibility: Martial Van Neste E

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 E
applications are written with a character based user interface. In fact, character- E
based devices are best suited for applications where the constraints of cost, speed, E
and the clutter of a pointing device on the desk are a major concern. E

It should be noted also that there are character-based window applications that E
may not have all the flexibility and ease of use of their graphic counterparts, but E
represent an alternative allowing the utilization of the large installed base of E
character terminals and still improve the ease of use. E

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 pre- E
clude the use of block-mode terminals, even though most applications built on E
POSIX -compliant platforms historically have used character-stream terminals. E

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.

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

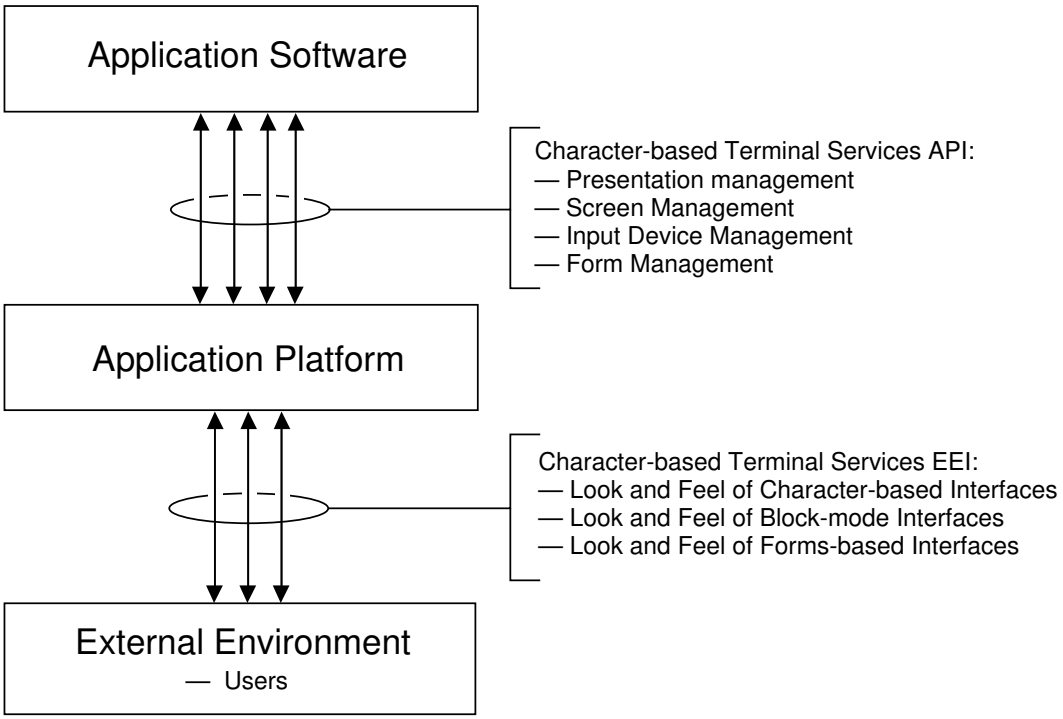

Figure 4-19 – Character-based Terminal Reference Model

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. E

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. E E E

These standard services support requirements for application portability and terminal independence. E E

Presentation Management E

Functions available for Presentation Management are: E

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

— Placement of text on the screen using a consistent reference E

— Positioning of the cursor for further output on the scree or for user input E

— Control of attributes of displayed text such as highlighting, underscoring, and coloring, if available E

— Clearing or refreshing the screen E

— Getting the current cursor position E

Screen Management E

Functions available for Screen Management are: E

— Control of the number and the width of the lines displayed E

— Use of a protected status line E

— Protection from writing or clearing in defined portions of the screen E

— Auto-wrapping in defined portions of the screen E

Input Device Management E

Functions available for Input Device Management are: E

— Configuration of the function keys, if available E

— Keyboard locking E

— Changing key mappings E

Form Management E

Functions available for Form Management are: E

— Definition of a form with different output and input text fields E

— Definition of the attribute input fields, such as text or different numeric formats E

— Generic and customizable error handling procedures for incorrect input E

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: E

— Style of selecting commands E

— Accessing online help E

— Performing common functions such as page forward and page backwards. E

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

— 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 E E

4.9.6 OSE Cross-Category Services

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

4.9.6.1 Security

To Be Provided. E

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. E E E

4.9.7 Related Standards

None.

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

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

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.

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

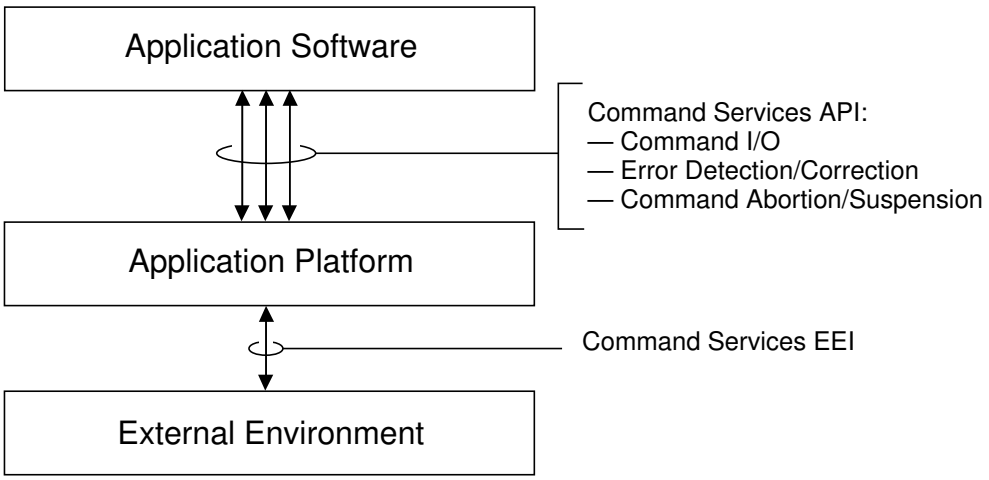

Figure 4-20 – POSIX OSE Reference Model for Command Interfaces

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.

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

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

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

3745 — Redirect the input for a command from a file

3746 — Direct the output of a command to be used as the input to another com-
3747 mand

3748 — Execute applications

3749 — Get online help for commands or applications

3750 — Manipulate file contents:

3751 • Cutting

3752 • Pasting

3753 • Concatenating

3754 • Converting

3755 • Sorting

3756 • Reformatting

3757 • Comparing

3758 • Searching for regular expression

3759 — Edit files

3760 • Interactive editors

3761 • Batch or ‘‘stream’’ editors

3762 — Display files

3763 • Pausing when necessary

3764 • Display only selected ranges of files

3765 — Manipulate files

3766 • Create

3767 • Delete

3768 • Rename

3769 • Move

3770 • Copy

3771 — Print files

3772 — Perform network functions

3773 • File transfer

3774 • Remote execution of commands

3775 • Remote file printing

3776 — Perform batch processing E

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

- 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

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

— 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
User Portability Extension (UPE)	E	IEEE POSIX. 2a	4.10.5.2
Control of interprocess communications, shared memory, and semaphores	E	IEEE POSIX. 4	4.10.5.2
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

4.10.5.1 Current Standards

There are no currently completed or approved international or national standards for commands and utilities.

4.10.5.2 Emerging Standards

IEEE POSIX. 2

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

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

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 E

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: E

— Utilities for submission and management of requests E

— System administration interfaces for the creation, management, and authorization of the network queueing and batch processing system E

— language-independent programmatic (library) interfaces for application access to utilities and the queue and request database, and E

— Application-level network protocols E

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: E

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

— 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 E standardized multilingual and multicultural support (e.g., localization requirements such as date formats and collation sequences, and support for international character sets).

4.10.7 Related Standards

None. E

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

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.

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

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