IEEE P1003.0 Draft 13 – September 1991

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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

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.

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.

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

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)

4.7.8 Open Issues

None.

4.7.9 Performance and Capacity Specification Metrics

Picture-Level Benchmark — (Developed by the GPC Committee)

The Picture-Level Benchmark (PLB), developed by the GPC (Graphics Performance Characterization) Committee, is a software package that provides a standard method of measuring graphics display performance for different hardware platforms. The GPC Committee consists of the following member companies: DEC, DuPont Pixel Systems, Evans & Sutherland, HP/Apollo, IBM, Intel, Integraph, Megatek, Prime Computer, Silicon Graphics, Sun, and Tektronix. NCGA (National Computer Graphics Association) is the administrator for the GPC work. The PLB is a part of ongoing work by the GPC Committee to develop standardized methods of measuring graphics performance.

The PLB was developed to clear the confusion regarding graphics performance measurement. Vectors or polygons per second can vary greatly according to size, orientation and other parameters. With the PLB, graphics display performance for the user's specific application can be evaluated using the same methodology and measurement criteria for different hardware systems. This allows ''apple-to-apple'' comparisons of graphics display performance.

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

4.8 Character-Based User Interface Services

Responsibility: Charles Severance

4.8.1 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.

This clause is one portion of the User Interface API and EEI as described in Section 3.

4.8.2 Scope

The scope of this clause is limited to the services and standards required to support character (non-bitmapped) terminals.

4.8.3 Reference Model

This subclause identifies the entities and interfaces specific to the character-based terminal services of an OSE.

As illustrated in Figure 4-18, 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.

4.8.4 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 similar between applications to make moving between applications as simple as possible.

4.8.4.1 Application Program Interface Services

The service requirements at the Application Program interface are as follows:

— Ability to support a wide variety of character-based terminals using a single terminal-independent API.

— Ability to support a wide variety of block mode terminals using a single terminal-independent API.

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

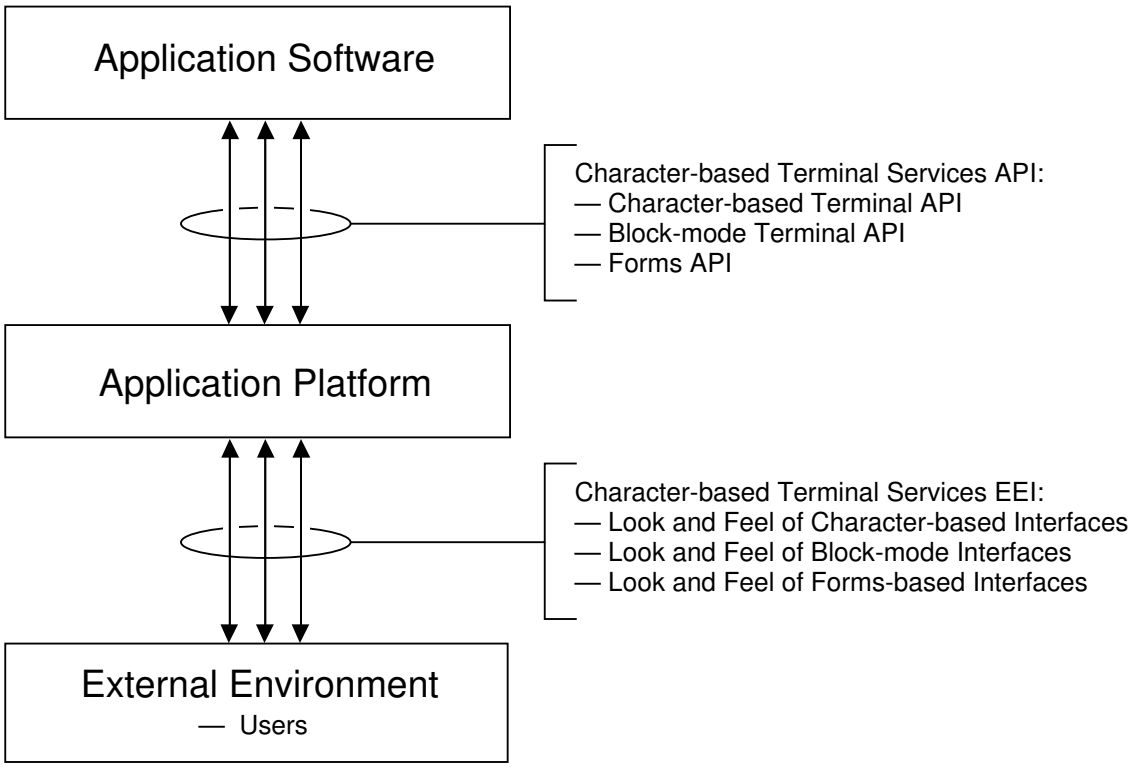

Figure 4-18 – Character-based Terminal Reference Model

— Ability to support all types of terminals using a forms-based API.

These interfaces should be complete enough to allow a wide range of applications to be implemented using the interfaces.

4.8.4.2 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:

— Placement of text on the screen

— Style of selecting commands

— Ways of using online help

— Ways to do common functions such as page forward and page backwards.

These interactions will differ slightly between different types of terminals because of limitations of the terminals.

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

4.8.4.3 Related Service Requirements

To be provided.

4.8.5 Standards, Specifications, and Gaps

4.8.5.1 Current Standards in the POSIX OSE

None.

4.8.5.2 Emerging Standards in the POSIX OSE

FIMS

ANSI CODASYL. Working draft 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.

CRS : Parts of POSIX. 2 might belong here.

4.8.5.3 Gaps in Available Standards

4.8.5.3.1 Standards/Specifications Outside the POSIX OSE

4.8.5.3.1.1 Government/Legal Standards

None.

4.8.5.3.1.2 De Facto Standards

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 standard satisfies some of the service requirements for character mode terminals.

SAA

SAA is an interface specification to a large class of block mode terminals. This standard satisfies some of the service requirements for controlling block mode terminals.

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

4.8.6 OSE Cross-Category Services

D

4.8.6.1 Security

4.8.6.2 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.

4.8.6.3 Configuration Management

4.8.6.4 Fault Management

4.8.6.5 Performance

4.8.6.6 Accounting

4.8.6.7 Training

4.8.6.8 Systems Integration Interface Requirements

None.

4.8.7 Related Standards

None.

4.8.8 Open Issues

None.

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

4.9 User Command Interface Services

Responsibility: Wendy Rauch

HLJ : Wendy has provided a full clause replacement for Draft 13; it is not further diff marked. D D

4.9.1 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.

4.9.2 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 the POSIX Open Systems Environment Guide.

4.9.3 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 4-19 shows, the command interface is available both to users (through the External Environment Interface) and to applications (through the Application

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

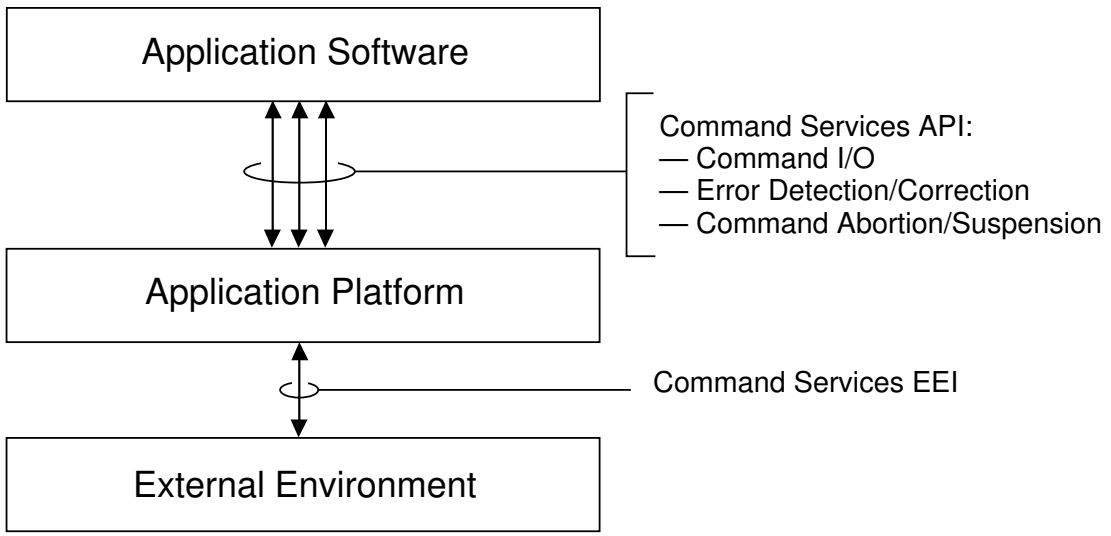

Figure 4-19 – POSIX OSE Reference Model for Command Interfaces

Programming Interface). Any operating system implementation can reside underneath the APIs and EEIs.

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.

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

4.9.4 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.

4.9.4.1 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 ''touchie-feelie'' screen, a tablet, or other input device.

The major services at the EEI provide the following abilities:

— Capture the output of a command or application into a file

— Redirect the input for a command from a file

— Direct the output of a command to be used as the input to another command

— Execute applications

— Get online help for commands or applications

— Manipulate file contents:

  • Cutting

  • Pasting

  • Concatenating

  • Converting

  • Sorting

  • Reformatting

  • Comparing

  • Searching for regular expression

— Edit files

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

- 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

— 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

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

— 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.

4.9.4.2 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.

4.9.4.3 Interapplication Entity Services

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

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

— 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.

4.9.5 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 4-10 summarizes the shell and utilities standards and specifications and work in progress.

Table 4-10 – Shell and Utilities Standards Activities

Service	Specification	Subclause
Shell and Utilities	IEEE POSIX. 2 {*ref*}	4.9.5.?
User Portability Extension (UPE)	IEEE POSIX. 2a {*ref*}	4.9.5.?
Shell, Utilities, and UPE	X/Open XPG 4.9.5.? OSF OSF /1 System V: 1989 SVID Berkeley BSD 4.x UNIX	

4.9.5.1 Current Standards

4.9.5.1.1 International and National Standards

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

4.9.5.2 Emerging Standards

4.9.5.2.1 International and Other Formal Standards

When completed, the IEEE POSIX. 2 {*ref*} 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}.

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

The IEEE POSIX. 2a {*ref*} User Portability Extension will supplement POSIX. 2 {*ref*} 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 {*ref*} 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 {*ref*} 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 {*ref*} and POSIX. 2a {*ref*} 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 {*ref*} and POSIX. 2a {*ref*} 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).

4.9.5.3 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.

4.9.5.3.1 Public Specifications

Public specifications that include the POSIX. 2 {*ref*} and POSIX. 2a {*ref*}, and go beyond these standards to also include the traditional UNIX-based command interfaces for software development, system administration, the UUCP (UNIX-to-UNIX Communications Protocol) utilities, and parts of the POSIX. 4 realtime extensions are available from a number of organizations. These include:

— X/Open's XPG3 specifications, Volume 1 and part of Volume 3

— OSF's OSF/1 Application Environment Specifications (AES), software documentation, and software

— University of California at Berkeley BSD UNIX

— AT&T System V Interface Definition (SVID) and System V UNIX.

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

4.9.6 POSIX OSE Cross-Category Services

4.9.6.1 Internationalization

The ANSI C, IEEE POSIX. 1, and IEEE POSIX. 2 specifications contain limited capabilities for obtaining and defining locale-specific information. In addition, some of the utilities described in the POSIX. 2 specifications contain requirements for standardized multilingual and multicultural support (e.g., localization requirements such as date formats and collation sequences, and support for international character sets).

4.9.7 Related Standards

Several standards activities are in progress to address additional commands interfaces that are traditionally included in UNIX and UNIX-like systems. The following standards are examples:

— IEEE 1003.4 Realtime Extensions for POSIX

— IEEE 1003.7 POSIX System Administration Extensions

— IEEE 1003.18 Platform Environment Profile (PEP) standard, which identifies and combines the standards dealing with POSIX. 1, POSIX. 2, POSIX. 2a, and the related standards

4.9.8 Open Issues

Conformance Testing

Command interface standards must address conformance testing. Although the POSIX. 2 standard is still in a draft stage, work has already begun in the IEEE 1003.3.2 Conformance Test Methods Group to define test assertions for the POSIX. 2 standard.

UUCP

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).

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

4.10 Transaction Processing Services D

Responsibility: Bob Gambrel

Based on input created by Carl Hall of P1003.11, with contributions from Jeremy Denny, John Penny, Patrick Prange, and Jeff DeMent Revised in Draft 13 with input from Dorm Fulton of P1003.11. D D

4.10.1 Overview and Rationale

The database management clause (see 4.4) described some transaction processing 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 and so are treated here as a separate clause. D D D D D D D D

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. D D D D D D D D

The following is quoted with a few editorial changes from ISO/IEC DP 10026-1 {*ref*}, 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.

D

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

4.10.2 Scope

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

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

— Transaction access to nondatabase ''resource managers'' (such as the software managing the cash in an automatic teller machine D

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

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

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

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

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. D

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

4.10.3 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. D

4.10.3.1 Conventional Transaction Processing Reference Model D

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

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

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 4-11. 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 4-20.) For convenience in discussing the model, the provider of those services is called the *Transaction Manager* (TM).

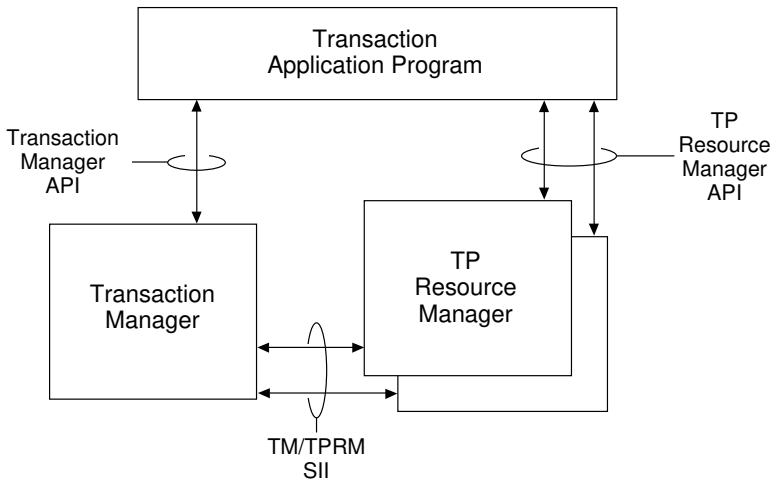

Figure 4-20 – The Conventional Transaction Processing Model

The transaction application program requests services provided by the *TP resource manager* (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. This illustration also shows the limits of transaction processing resource management. The machine could signal completion, but a mechanical problem could

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

prevent the cash from being dispensed correctly, undetected by the system.

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

4.10.3.2 POSIX OSE Reference Model (with Transaction Processing)

The conventional transaction processing model is shown integrated into the D
POSIX OSE Reference Model in Figure 4-21. 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 ser- D
vices APIs and EEIs. D

4.10.3.3 Real World Considerations

The POSIX OSE Reference Model does not provide for a way to expose the details
of the Application Platform. In the Transaction Processing world, as shown in the
conventional Transaction Processing Reference Model (see 4.10.3.1), the existence
of Transaction Managers and multiple TP Resource Managers connected by the D
TM/TPRM SII is important. One way to think about the real world implications of D
this is that TP Resource Managers and the Transaction Managers could both be D
implemented in the Application Platform as separate entities, connected to each D
other by the TM/TPRM SII. This does not, however, imply that the two must be D
implemented as separate entities, though there are advantages to the user if they D
are separate. D

4.10.4 Service Requirements

Services provided via the Transaction Processing Services API are described in D
4.10.4.1. Services to enable the distribution of transaction processing are
described in 4.10.4.2. General services, mostly performing administrative func-
tions, are described in 4.10.4.4. Services provided across the TM/TPRM SII are D
briefly described in 4.10.4.5. D

4.10.4.1 Application Program Interface Services

This subclause describes the major categories of services available at the POSIX D
API. D

The Transaction Services API provides various services to the application pro- D
grammer: D

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

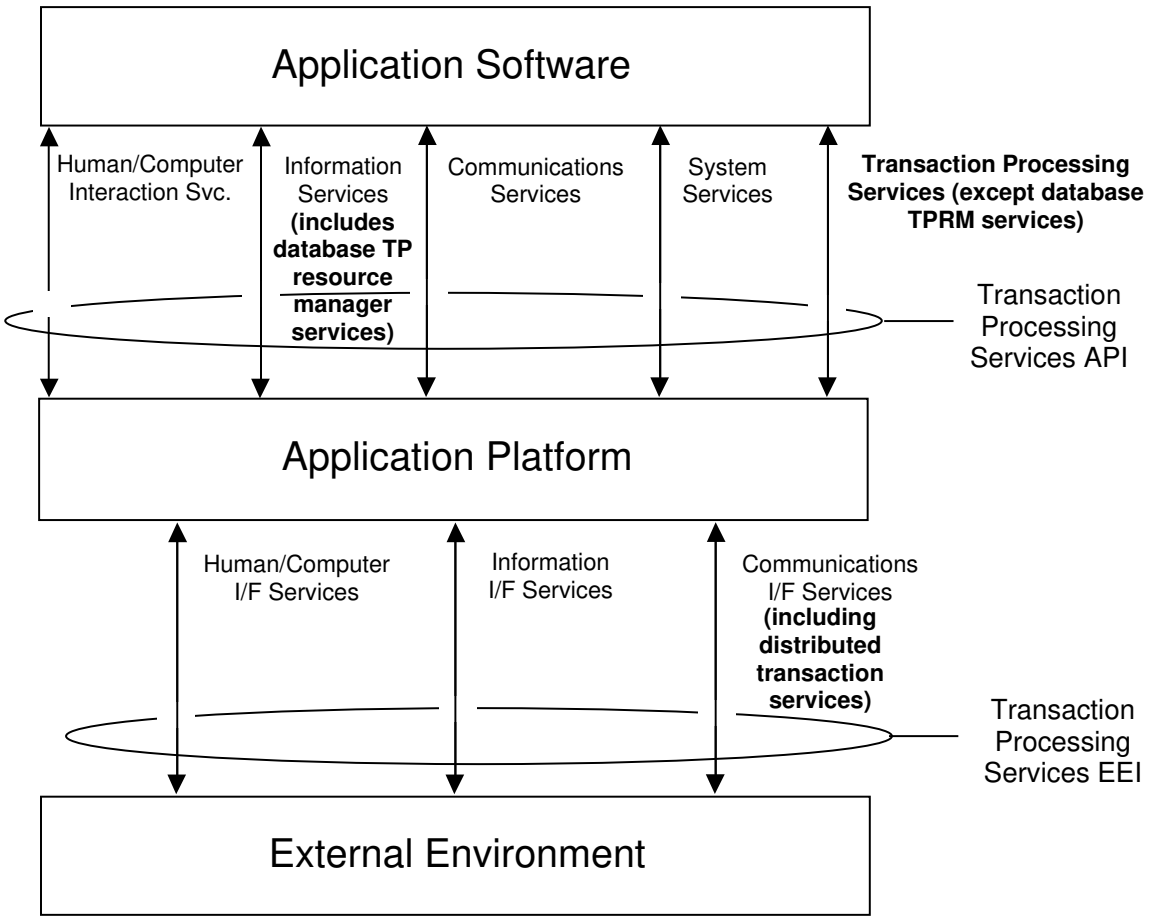

Figure 4-21 – The POSIX OSE Transaction Processing Reference Model D

Transaction Demarcation D

Ability to indicate the start of a transaction. D

Ability to indicate a transaction has ended successfully (commit) or unsuccessfully (rollback). D D

Ability to 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.) D D D D D D

The ability to 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. D D D

Ability to explicitly include or exclude items from a transaction. *RJG : PO—this was yours. Isn't this really the above nesting item?* D D

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

Communications Between Transaction Application Programs

Ability to call another transaction application program (possibly remote) within the context of a transaction.

Ability to 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.'' The above services often go by shorthand names such as ''Remote Procedure Call (RPC),'' ''Single Request/Response,'' or ''Client/Server,'' in the first case, and ''Dialogs,'' ''Conversations,'' or ''Peer-to-Peer'' in the second case.

Terminal Communications

Ability to 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

Ability to 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

Ability to 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.

Ability to ''send'' a message to another transaction application program.

Ability to retrieve the next message (and then act upon it)

Ability to 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 covered earlier in this document.

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

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